Centro Universitário de Brasília.
Faculdade de Ciências Exatas e Tecnologia.

Projeto Final de Graduação do Curso de Engenharia da Computação

Comparação de preços via celular, utilizando reconhecimento de código de barras

Por

João Paulo Galvagni Júnior

R.A.: 2001593/4

Brasília, 2º Semestre de 2006.

Centro Universitário de Brasília.
Faculdade de Ciências Exatas e Tecnologia

Projeto Final de Graduação do Curso de Engenharia da Computação

Comparação de preços via celular, utilizando reconhecimento de código de barras

Por

João Paulo Galvagni Júnior

R.A.: 2001593/4

Orientador:
Claudio Penedo de Albuquerque

Brasília, 2º Semestre de 2006.

DEDICATÓRIA

Dedico a Deus, pela inspiração, à minha família pelo apoio e incentivo e à Lívia Linhares Santiago Santos, pelo estímulo e suporte na realização deste trabalho.

Dedico-lhes essa conquista como gratidão.

AGRADECIMENTOS

Agradeço primeiramente a Deus pelo dom da vida e pelas oportunidades a mim concedidas durante minha vida.

Ao meu mestre e professor Claudio Penedo, que me ofereceu a devida orientação para o bom desenvolvimento de meu projeto.

RESUMO

Os códigos de barras são utilizados para padronizar os produtos comerciais e industriais, facilitando o controle a nível nacional e internacional. Os leitores de códigos de barras já estão presentes na maioria dos estabelecimentos comerciais, sendo de grande valia nos caixas e controle de estoque.

O objetivo deste trabalho é simular aos usuários uma ferramenta capaz de auxiliar na economia doméstica de forma prática e eficiente. Para tal, especifica-se e implementa-se o projeto de um *software* completo capaz de realizar uma comparação de preços de produtos em diversos estabelecimentos e, baseando-se nestas pesquisas, são retornados os resultados mais satisfatórios para o usuário. O fluxo do processo terá a duração de segundos.

Na implementação deste projeto, empregar-se-á a tecnologia de reconhecimento de código de barras, devido à necessidade de leitura e de interpretação dos códigos de barras dos produtos, bem como a tecnologia móvel celular como interface entre a aplicação e o usuário final. O padrão de código de barras utilizado no escopo desse projeto será o mundialmente conhecido EAN-13.

Palavras-chave: Reconhecimento de código de barras, EAN-13, comparação de preços, tecnologia móvel celular.

ABSTRACT

The barcodes are used to standardize the commercial and industrial products, facilitating a control at national and international level. The barcode readers already are present at the majority or the commercial establishments, being of great value at cash registers and supply control.

The purpose of this work is to simulate to users one tool capable to assist at domestic economy of practical and efficient way. For such, it is specified and implemented a project of complete software capable to carry through a comparison between prices of products of diverse establishments, and basing on this search, return the more satisfactoriness results to user, such as the cellular mobile technology as the interface between the application and the final user. The flow of the process will have the duration of seconds.

On the implementation of this project, it will be used technology the bar code recognition, due to necessity of reading and interpretation of the bar codes of the products. The standard to be used in the scope of this project will be the worldwide known as EAN-13.

Keywords: Bar code recognition, EAN-13, price comparison, cellular mobile technology.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

API – Application Programming Interface

DAO – Data Access Object

DP – SmartTrust Delivery Platform

API – Application Programming Interface

DAO – Data Access Object

DP – SmartTrust Delivery Platform

DTD – Document Type Definition

EAN – European Article Numbering

EAN.UCC – European Article Numbering – Uniform Code Council

ETSI – European Telecommunications Standards Institute

GS1 – The First in Global Standards

GSM – Global System for Mobile Communications

J2ME – Java 2 Micro Edition

JDBC – Java Database Connectivity

JSP – Java Server Page

MSISDN – Mobile Station International ISDN Number

OTA – Over The Air

Plataforma OTA – Plataforma Over The Air

SGBD – Sistema de Gerenciamento de Banco de Dados

SM – Short Message

SMS – Short Message Service

SMS-BC – Short Message Service – Cell Broadcast

SMSC – Short Message Service Centre

SMS-PP – Short Message Service – Point to Point

URL – Universal Resource Locator

WAC – WIG Application Creator

WAP – Wireless Application Protocol

WIB – SmartTrust WIB™

WIG – Wireless Internet Gateway

WML – Wireless Markup Language

XML – eXtensible Markup Language

Capítulo 1 - Introdução

Este trabalho trata sobre o tema de consulta e comparação de preços, utilizando a tecnologia de reconhecimento de código de barras como solução e a rede de telefonia móvel como meio de comunicação dos dados.

1.1 - **Motivação**

A evolução da tecnologia móvel celular tem permitido o surgimento de novos serviços e utilidades quase que diariamente. Essas novas tecnologias visam facilitar as tarefas do dia-a-dia e, de uma maneira geral, tornam a utilização de celulares cada vez mais imprescindíveis no cotidiano da população.

Uma das razões para a escolha do tema foi a possibilidade de disponibilizar um serviço simples e útil ao alcance de todos os usuários de telefonia móvel. E acompanhando essa evolução tecnológica, na qual se baseia o tema, nada mais útil e econômico do que agregar um serviço como esse a uma tecnologia já bastante difundida entre a população. Isso pode ser verificado até mesmo pela competitividade existente no mercado atual de telefonia móvel, pois as empresas acompanham as tendências tecnológicas mundiais, presentes nos países de primeiro mundo.

Muito mais do que mesclar os conhecimentos da telefonia móvel com a tecnologia de reconhecimento de código de barras, esse projeto torna-se uma ótima fonte de estudos e investimento profissional.

1.2 - **Proposta do Projeto**

Este projeto tem como objetivo o desenvolvimento de um sistema capaz de pesquisar os preços de produtos comerciais em tempo real, comparando-os com os produtos de outros estabelecimentos, visando obter uma economia significativa com isso.

Para isso será utilizado, além da tecnologia de telefonia móvel, o reconhecimento de código de barras. Tal tecnologia já se encontra presente na maioria dos estabelecimentos comerciais que conhecemos, como restaurantes, supermercados, catracas, entre outros, sendo que sua utilização está se popularizando e se tornando, dessa forma, uma alternativa de baixo custo para a leitura óptica e padronização de produtos e serviços.

Uma base de dados conterá os preços dos produtos e informações dos estabelecimentos que o comercializam, sendo a fonte da comparação entre os preços. A base de dados estará situada no mesmo ambiente do *software*, o que facilita a comunicação entre os sistemas. Será necessário que o código de barras esteja em um formato numérico, para que o mesmo possa ser armazenado no banco de dados.

Uma fotografia digital do aparelho celular é o ponto inicial para que o usuário interessado na comparação de preços obtenha o resultado esperado. Em seguida, o usuário deverá enviar a imagem para a aplicação responsável pela “leitura” e “interpretação” dessa imagem, gerando precisamente o código de barras no formato numérico, semelhante aos códigos cadastrados na base de dados.

Depois de realizada uma consulta ao banco de dados, o usuário recebe uma mensagem na tela de seu aparelho, contendo um comparativo entre os menores preços e seus respectivos estabelecimentos, respeitando a localidade em que se encontra.

Na Figura 1.1, ilustra-se a fotografia de um código de barras, retirada com um celular, no padrão EAN-13, que será utilizado no escopo do proposto projeto.

Figura 1.1 – Fotografia digital de um código de barras

Como alternativa aos usuários que não possuem aparelho celular com câmera digital embutida, será disponibilizada uma interface amigável através do WIB *Browser*, conhecida como "menu de serviços" do simcard, onde o usuário poderá digitar o código de barras já em seu formato numérico. Os passos seguintes são semelhantes ao descrito anteriormente.

1.3 - **Estrutura do Trabalho**

O Capítulo 1 tem como objetivo introduzir os conceitos a serem utilizados durante o desenvolvimento do trabalho em questão, bem como a estruturação do mesmo.

O Capítulo 2 descreve os códigos de barras, sua história e suas características, algo introdutório para a proposta que se segue.

O Capítulo 3 descreve as tecnologias utilizadas no desenvolvimento do projeto, as ferramentas de simulação e testes e as linguagens utilizadas no trabalho. Inclui ainda o Banco de Dados, com detalhamento da tecnologia e estrutura das tabelas.

No Capítulo 4 será descrito o processo de reconhecimento de código de barras, bem como do *software* de gerenciamento das consulta ao banco de dados. Nesse Capítulo será descrito também a alternativa de utilização do serviço para aparelhos que não possuem câmera digital.

O Capítulo 5 trata da conclusão obtida ao final do trabalho e, com base nas tecnologias e tendências de mercado, há sugestões para trabalhos futuros derivados deste.

Capítulo 2 - Código de Barras

Como tecnologia a ser utilizada no escopo deste trabalho é a de reconhecimento de código de barras, é necessário o detalhamento de alguns conceitos ligados ao código de barras como histórico, padrões, entre outros. O Capítulo 2 irá descrever tais conceitos e explicar como esses conceitos foram utilizados no desenvolvimento do presente trabalho.

2.1 - **Histórico**

Em novembro de 1983, surge no Brasil a Associação Brasileira de Automação Comercial – ABAC, posteriormente conhecida como EAN BRASIL, com o intuito de assegurar a legitimidade e a legalidade do processo de automação comercial. [**EAN BRASIL – 2006**]

Em junho de 1984, durante o primeiro congresso em nível nacional, a ABAC chegou à conclusão de que a implantação do Código Nacional de Produtos (código de barras) no Brasil era primordial. Após ter sido apresentada ao governo, tal recomendação foi transformada em lei (Decreto 90.595, de 29 de novembro de 1984) e em dezembro de 1984 era publicada a Portaria 143 do Ministério da Indústria e Comércio, conferindo à ABAC a responsabilidade de orientar e administrar a implantação do Código Nacional de Produtos no país. [**EAN BRASIL – 2006**]

Para a viabilização deste projeto, fez-se necessária a adoção de um padrão internacional. Assim sendo, optou-se pelo padrão EAN Internacional. Na condição de 26º país filiado, o Brasil recebeu seu número de código, a flag 789, em maio de 1985. [**EAN BRASIL – 2006**]

Com o objetivo de fortalecer a imagem da Entidade nos diversos campos de atuação, a Associação Brasileira de Automação Comercial mudou sua sigla de ABAC para EAN

BRASIL, pois se tornou necessário desvincular a sigla de um único segmento e tornar a associação reconhecida nacional e internacionalmente, como representante legal responsável pela utilização dos Sistemas EAN no Brasil. Sendo assim, a GS1 Brasil (nova marca da EAN BRASIL, em uma referência à "*The first in Global Standards*" – a Primeira em Padrões Globais) tem atuado com o objetivo de estabelecer normas técnicas necessárias, promover a cooperação entre parceiros comerciais, assegurar apoio aos empresários, divulgando para isso novas tecnologias e, principalmente, incentivando a modernização. A transição ocorreu oficialmente em fevereiro de 2005, no âmbito global. [**EAN BRASIL – 2006**]

Dessa forma, a GS1 Brasil chega ao Brasil com a seguinte missão:

"Criar e implementar padrões globais para melhorar a cadeia de suprimentos e de demanda". [**EAN BRASIL – 2006**]

2.2 - **O Sistema EAN.UCC**

O sistema EAN.UCC, por ser um conjunto de padrões, possibilita a gestão eficiente de cadeias de suprimentos globais e multissetoriais, identificando com exclusividade produtos, unidades logísticas, localizações, ativos e serviços. Tem por objetivo facilitar os processos de comércio eletrônico, propondo soluções estruturadas para mensagens eletrônicas e viabilizando a total rastreabilidade das operações. [**EAN BRASIL – EAN.UCC**]

Os números de identificação podem ser representados por meio de símbolos do código de barras, possibilitando a leitura eletrônica (óptica) em qualquer etapa, desde o ponto de venda, recebimento, até os depósitos, onde será necessária a captura de dados nos processos de negócios. Projetado para superar as limitações decorrentes do uso de codificações específicas (restritas) de um setor, empresa ou organização, o Sistema torna o comércio muito mais eficiente e ativo aos clientes. [**EAN BRASIL – EAN.UCC**]

O Sistema proporciona, além de números exclusivos de identificação, informações adicionais como números de série, datas de validade e números de lote mostrados na forma de código de barras. [**EAN BRASIL – EAN.UCC**]

2.3 - **Sistema de Numeração**

Com o constante crescimento do comércio global e do uso de computadores, sistemas de identificação capazes de ser usados em todos os setores da indústria e comércio mundial devem substituir as descrições de produtos e serviços em linguagem simples. Para atender a essa necessidade, fornecendo soluções que garantam identificação exclusiva e sem ambigüidades é que surge o Sistema EAN.UCC. Comerciantes, exportadores, importadores, hospitais, fabricantes, varejistas, e vários outros podem utilizar o sistema para comunicação relativa aos serviços e mercadorias que comercializam. [**EAN BRASIL – Sistema de Numeração**]

A representação dos números de identificação exclusivos é feita através de símbolos de código de barras, o que possibilita a captura de dados precisa e com baixo custo, fornecendo assim as informações necessárias em todos os pontos da cadeia de suprimentos. [**EAN BRASIL – Sistema de Numeração**]

2.4 - **Estrutura do código de barras**

Dos diversos códigos de barra existentes no Brasil e no mundo, o EAN13 (ou EAN.UCC-13) será o padrão utilizado no escopo desse projeto, pois é o código mais utilizado na identificação de itens comerciais e é ilustrado na Figura 2.1.

Figura 2.1 – Exemplo de um código de barras EAN.UCC-13

O código de barras em questão é formado por 13 dígitos, conforme ilustra a Figura 2.2.

Posição	1	2	3	4	5	6	7	8	9	10	11	12	13
Exemplo	7	8	9	8	3	5	7	4	1	0	0	1	5

Figura 2.2 – Detalhe de um código no padrão EAN.UCC-13

Os 3 primeiros dígitos servem para identificar o país, no caso do Brasil o código é 789, os 4 dígitos seguintes determinam a empresa detentora do produto/serviço filiada à EAN, os próximos 5 dígitos representam o código do item comercial dentro da empresa. De maneira geral, a combinação entre o código da empresa e o código do item deve ter 9 dígitos, podendo ambos variar de 3 a 6 dígitos em sua composição. O 13º é o dígito verificador, obtido através de um cálculo matemático, mostrado na seção seguinte.

2.4.1 - Cálculo do Dígito Verificador

Para que um código de barras seja válido, o dígito verificador tem que obedecer a um cálculo. Abaixo serão mostradas as etapas desse cálculo, validando o dígito verificador para o código de barras da Figura 2.1, por exemplo.

Passo 1: Considerando apenas os 12 primeiros dígitos, faz-se uma associação de posição e, em seguida, multiplicam-se os números das posições pares ao peso 3 (três) e das posições ímpares ao peso 1 (um).

Código:	7	8	9	8	3	5	7	4	1	0	0	1
Peso:	1	3	1	3	1	3	1	3	1	3	1	3

Passo 2: Multiplica-se cada dígito pelo seu respectivo peso e somam-se todos os resultados dessa multiplicação.

$$7 + 24 + 9 + 24 + 3 + 15 + 7 + 12 + 1 + 0 + 0 + 3 = \mathbf{105}$$

Passo 3: O resultado dessa soma deve ser subtraído do múltiplo imediatamente superior a ele, múltiplo de 10, nesse caso o número 110.

$$110 - 105 = \underline{5}$$

Caso a soma total seja um múltiplo de 10, o dígito verificador será 0 (zero).

Conclusão: O dígito verificador da seqüência anterior, conforme a imagem do código de barras, é o 5 (cinco). Portanto, o código completo resultante é: **789 8357 41001 5**.

Esse cálculo está contemplado em uma classe específica para a validação do dígito verificador, agregada ao *software* de reconhecimento de código de barras, o que garante a consistência dos códigos a serem consultados na base de dados.

Capítulo 3 - Ferramentas Tecnológicas

Antes de dar seqüência na simulação do sistema, faz-se necessário o esclarecimento de alguns conceitos que serão aplicados durante o processo de consulta e desenvolvimento da solução.

Neste capítulo serão abordadas as principais tecnologias utilizadas no projeto, bem como a explicação e o funcionamento das ferramentas.

3.1 - **Plataforma *Over The Air* (*OTA*)**

A Plataforma *OTA* (ou Sobre o Ar) é um conjunto de ferramentas proprietárias da SmartTrust, cujo principal objetivo é fornecer serviços e aplicações específicas para terminais móveis e chips *GSM*, conhecidos como simcard.

Dentre as tecnologias que a Plataforma *OTA* abrange apenas algumas serão utilizadas no decorrer do projeto. São elas:

- *Delivery Platform* – Plataforma de Entrega
- *WIG Server* – Servidor WIG
- *Transport Server* – Servidor de Transporte
- SmartTrust *WIB*

3.2 - ***Delivery Platform***

A DP foi desenvolvida para instalar, gerenciar e reparar aplicações e serviços no simcard e nos aparelhos celulares, sem a necessidade de uma ligação física entre o dispositivo e o servidor. Com esse módulo, a *OTA* é capaz de adicionar, atualizar e/ou remover serviços contidos nos simcards de forma rápida e mantendo a consistência entre os serviços no simcard e na base de dados que controla a distribuição dos mesmos. A Figura 3.1 ilustra a

comunicação entre o aparelho celular e a plataforma *OTA* utilizada no escopo do presente trabalho.

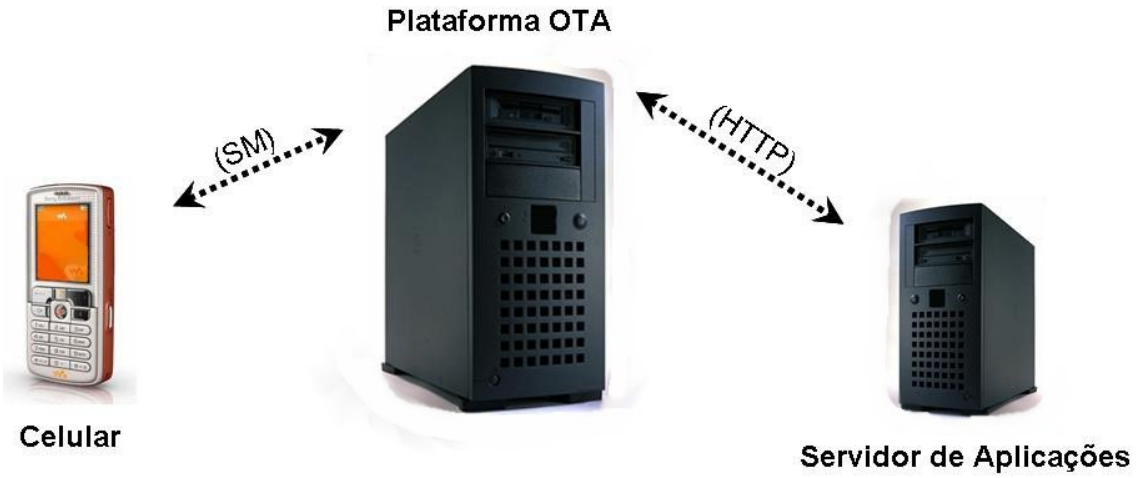

Figura 3.1 – Estrutura da Plataforma *OTA* e sua comunicação

As informações desse módulo passam por um *Gateway*, que faz o trabalho de conversão dos tipos de informações trafegados entre o servidor e o dispositivo móvel celular. Outro módulo que participa no tráfego dessas informações é a Plataforma SMSC, conhecido como Centro de Serviço de Mensagens de Texto. Essa serve apenas para transportar as mensagens da Plataforma *OTA* para o dispositivo e vice-versa.

No que diz respeito à Plataforma *OTA*, podem-se destacar duas camadas principais:

- *WIG Server*:

Trabalha como um *Gateway* bidirecional, ou seja, tem como principal função o envio de requisições para os provedores de conteúdo na internet, bem como envio de mensagens para os aparelhos celulares, chamadas *Push*[1] [**WIKIPEDIA - *WAP PUSH***]. Dessa forma, a principal tarefa do *WIG Server* é receber códigos *WIG WML* via requisições HTTP[2] [**WIKIPEDIA - HTTP**] dos provedores de conteúdo, convertê-los em código compreensível

[1] O Push é uma variação do *WAP PUSH*, definido como uma mensagem enviada diretamente ao celular, sem que esse tenha feito uma requisição pela mesma.

[2] HTTP (*HyperText Transfer Protocol*, ou Protocolo de Transferência de Hipertexto): protocolo utilizado para transferência de dados na rede mundial de computadores.

pelos *WIB* e enviá-los ao dispositivo celular através do *Transport Server*. Na Figura 3.2, uma ilustração do princípio de funcionamento do *WIG Server*.

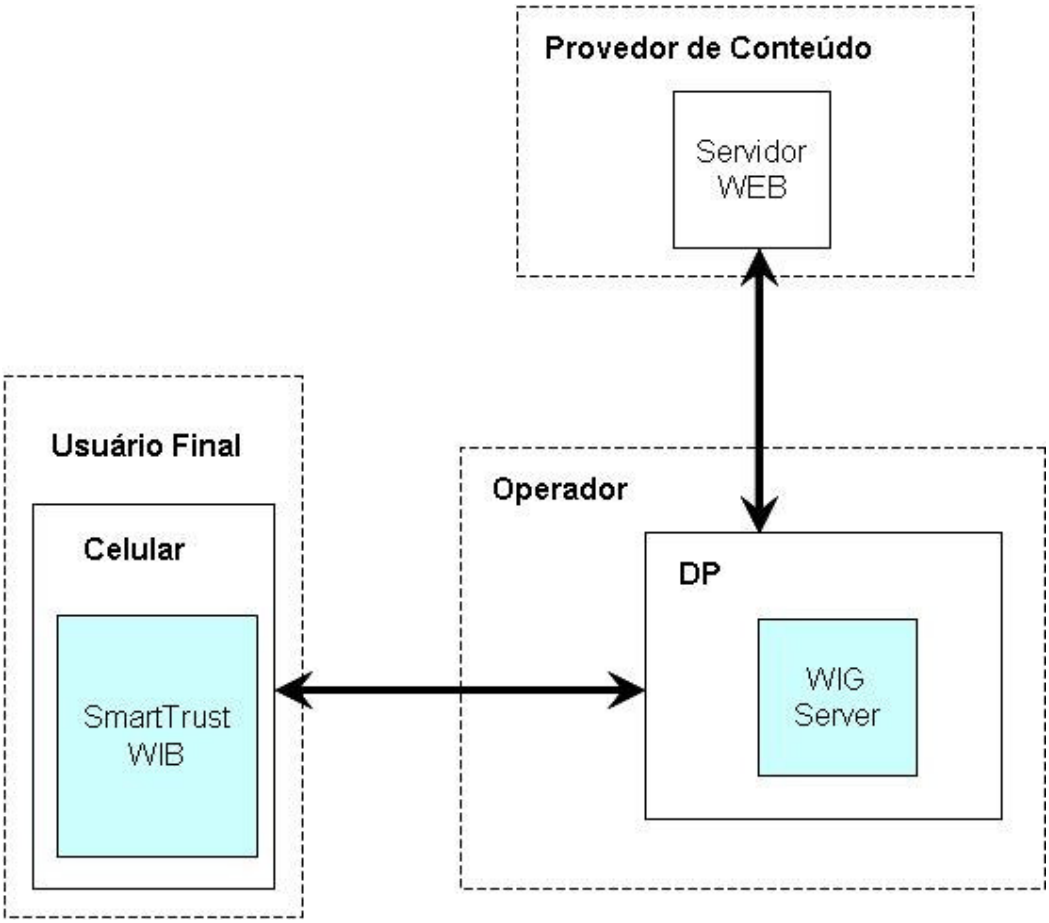

Figura 3.2 – Princípios do *WIG Server*

- *Transport Server*:

O *Transport Server* inclui, entre outras funcionalidades, a recepção de mensagens curtas de qualquer servidor, o envio de mensagens aos aparelhos celulares através da *SMSC*, gerenciar as mensagens enviadas através de status – mensagem enviada, a caminho e entregue, bem como status de erros. Como o *Transport Server* utiliza a interface da *SMSC*, seu serviço fica limitado às configurações e restrições.

O servidor de transporte é capaz de realizar o balanço de cargas através da criação de novas instâncias para dividir o tráfego entre as mesmas. Possui também um controle de

overflow[3] [**WENZEL, FLAVIO**], ou seja, é capaz de recusar conexões excedentes, garantindo assim um bom nível de serviço.

3.3 - **SmartTrust *WIB***

O *WIB – Wireless Internet Browser* – é um browser previamente instalado nos simcards que permite aos usuários acessar outras aplicações na internet. A escolha pela utilização do SmartTrust *WIB*™[4] se deu pela parceria já existente entre a SmartTrust e fornecedores de simcard, aliado ao fato de que esse produto já é utilizado por diversas empresas de telecomunicações em vários países.

O *WIB* é baseado em menus e sub-menus, utilizando a linguagem *WML* como padrão de código e, portanto, capaz de realizar tarefas simples como requisições à provedores de conteúdo ou servidores de aplicações e tarefas complexas como alterar parâmetros do simcard, acessar dados contidos no mesmo e enviar esses dados pela rede de telefonia móvel. Por essa razão é que são desenvolvidas diversas aplicações em servidores externos, para que os mesmos executem tarefas mais complexas e diversificadas. Dos diversos serviços existentes no mercado baseados em *WIB*, destacam-se as salas de bate-papos virtuais, conhecidos como *chat*[5] [**WIKIPEDIA - CHAT**], serviços bancários em geral, comércio de ringtones[6] [**WIKIPEDIA - RINGTONES**], entre outros.

No presente trabalho, a utilização do *WIB* terá como finalidade mostrar os resultados da consulta ao usuário do serviço, bem como disponibilizar a opção alternativa para aparelhos que não possuem câmera digital, conforme descrito no Capítulo 4.

---

[3] Conhecido entre os programadores, o estouro de pilha é quando há um limite máximo definido e o mesmo é ultrapassado, prejudicando o bom funcionamento do sistema como um todo.

[4] SmartTrust WIB™ é um browser proprietário da SmartTrust.

[5] Chat, que em português significa "conversação", é um termo que determina as aplicações de conversação em tempo real na web ou dispositivos móveis.

[6] Som emitido pelos aparelhos celulares para indicar uma chamada recebida.

Na Figura 3.3, uma ilustração de um Menu contido no simcard e como o mesmo é visualizado pelos usuários.

Figura 3.3 – Tela inicial do Menu Alternativo

3.4 - **Comunicação dos Dados**

Nessa seção serão abordadas algumas das principais tecnologias utilizadas na comunicação dos dados durante as etapas do sistema. A descrição de algumas tecnologias se limita à utilização das mesmas no projeto. Portanto, informações não pertinentes ao projeto, bem como informações confidenciais não serão abordadas nesse documento, visando manter a ética e evitar quaisquer intervenções aos interesses das empresas proprietárias.

3.5 - ***SMS***

O *SMS* – ***Short Message Service***, ou serviço de mensagens curtas já era discutido em meados do ano de 1980, porém a primeira mensagem de texto só veio a ser enviada em 1992. A primeira proposta era a utilização dessas mensagens para tráfego de informações de sinalização e controle. Porém a disseminação desse serviço entre as empresas de telecomunicações tornou o envio de mensagens algo tão comum quanto uma simples ligação

telefônica, tornando-se assim um dos principais itens de receita de valor agregado para as empresas.

São dois os tipos de serviços de mensagens existentes:

- *SMS*-PP (*Short Message Service – Point to Point*): É o envio de mensagem de um terminal móvel diretamente para outro, de forma singular. [**WIKIPEDIA - *SMS***]
- *SMS*-CB (*Short Message Service – Cell Broadcast*): É o envio de mensagem para vários terminais móveis dentro de uma determinada área geográfica. [**WIKIPEDIA - *SMS***]

Tecnicamente, as mensagens são enviadas para uma plataforma de controle de mensagens, cujo mecanismo de *Store and Forward* recebe e encaminha essas mensagens ao destinatário. São realizadas algumas tentativas de envio para o destino, que caso não esteja disponível por alguma razão, a mensagem será descartada após um período pré-determinado de expiração das mensagens. [**WIKIPEDIA - *SMS***]

A utilização das mensagens de texto dentro do proposto projeto é o principal meio de comunicação entre o aparelho celular e o servidor de aplicações, passando através da *SMSC* e da Plataforma *OTA*. A *SMSC* apenas fará a entrega das mensagens para os destinatários, ora o aparelho celular, ora a Plataforma *OTA*, que por sua vez irá tratar cada mensagem de acordo com o seu propósito.

3.6 - *WIG WML*

A linguagem utilizada no projeto no que diz respeito ao aparelho celular é o *WIG WML – Wireless Internet Gateway Wireless Markup Language*. A linguagem *WML* é mundialmente conhecida e utilizada, porém para o presente trabalho será utilizada uma linguagem "adaptada", ou seja, a linguagem *WIG WML* proprietária da SmartTrust™. A linguagem em questão se assemelha muito à linguagens como *HTML* (*HyperText Markup Language*) e XML (*eXtensible Markup Language*), que são linguagens de marcação escritas

em códigos que podem ser interpretados pelos navegadores de internet para exibir páginas da Rede Mundial de Computadores (*WWW – World Wide Web*) [**WIKIPEDIA - HTML**]. No caso do *WML*, o mesmo é interpretado por um micro-navegador instalado em dispositivos móveis – celulares e PDAs.

A utilização do *WML* na aplicação se dá na interface dos usuários de aparelhos celulares sem câmera digital, disponibilizando um menu interativo para a consulta e comparação de preços. Da Figura 3.4 à Figura 3.7, um exemplo simulado do fluxo básico, demonstrando a interface real de comunicação com os usuários.

Figura 3.4 – Tela inicial do Menu Alternativo

Figura 3.5 – Opção “Cons. Preços” do Menu Alternativo

Figura 3.6 – Opção "Sobre" do Menu Alternativo

Figura 3.7 – Opção "Ajuda" do Menu Alternativo

As opções da Figura 3.4 são meramente ilustrativas, utilizadas para demonstrar como funciona a interação entre o usuário do serviço com o aparelho celular, abrangendo ainda os menus auxiliares "Ajuda", explicando como o usuário deve proceder para a realização de uma consulta e "Sobre", abrangendo o objetivo do serviço em questão. A codificação das telas mostradas acima são simples e são detalhadas no Apêndice A.

3.7 - **Java**

Uma linguagem simples e robusta, conhecida por ser multiplataforma, ou seja, independe do sistema em que está operando para executar suas aplicações e tarefas, bastando para isso uma *virtual machine*[7] [**WIKIPEDIA – VIRTUAL MACHINE**] instalada no Sistema Operacional. Essa é a linguagem Java™, a mais conhecida e utilizada linguagem orientada a objetos no mundo.

A tecnologia Java foi criada como parte de um projeto anônimo e secreto chamado "*The Green Project*" (ou "O Projeto Verde") da Sun Microsystems em 1991. Um grupo

[7] Em termos gerais, uma máquina virtual, como é conhecida em português, é um *software* que tem por objetivo criar um ambiente entre a plataforma – Sistema Operacional e o *software* do usuário final.

formado por 13 pessoas e liderado por James Gosling trabalhou durante 18 meses isolados em um escritório anônimo, longe de qualquer comunicação com o mundo exterior. A apresentação pública da tecnologia foi realizada em 1995. [**JAVA - BR**]

Após quase 10 desde o seu lançamento, a plataforma Java já possui mais de 4 milhões de desenvolvedores de *software* e é usada nos principais setores de tecnologia no mundo, estando presente em diversas redes de tecnologias de programação, dispositivos e computadores. [**JAVA - BR**]

A escolha da linguagem Java se deu, entre outros motivos, pela versatilidade e robustez que a mesma apresenta, após anos de amadurecimento, testes e atualizações realizadas por toda a comunidade ativa dos mais de quatro milhões de desenvolvedores. [**JAVA - BR**]

3.8 - **J2SE**

Para o desenvolvimento do presente trabalho foi utilizada a API[8] Java em sua versão 1.4, cuja tecnologia foi essencial e necessária para o funcionamento do proposto projeto. Na Figura 3.8, uma visão geral da estrutura do Java™ 2 *Platform*, *Standard Edition* versão 1.4.

[8] "*Application Programming Interface (ou Interface de Programação de Aplicativos) é um conjunto de rotinas e padrões estabelecidos por um software para utilização de suas funcionalidades por programas aplicativos – i.e.: programas que não querem se envolver em detalhes da implementação do software, mas apenas usar seus serviços*" [**WIKIPEDIA - API**].

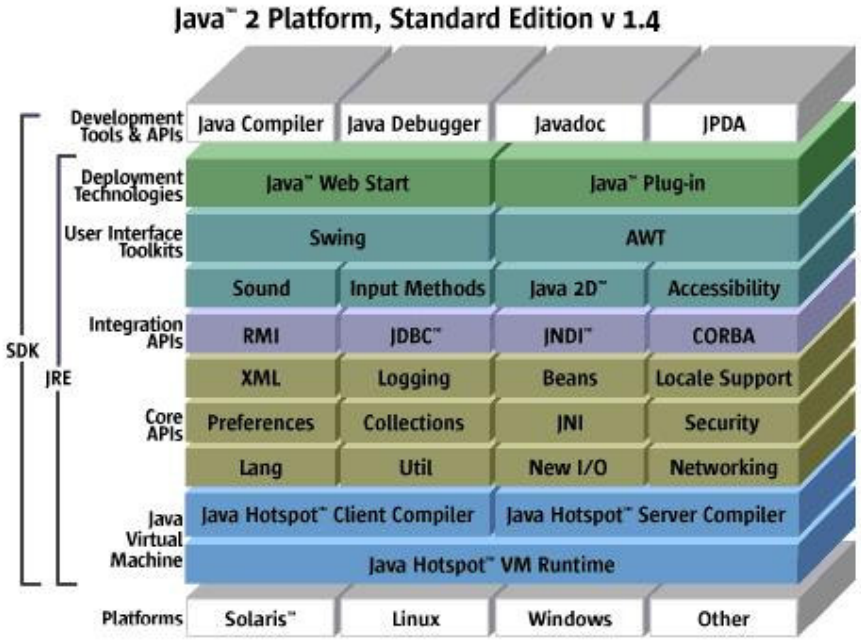

Figura 3.8 – Estrutura Geral da Plataforma Java

Como é observado na Figura 3.8, a linguagem Java é capaz de rodar seus aplicativos em vários sistemas operacionais, bastando apenas possuir a máquina virtual instalada. Os módulos utilizados no desenvolvimento do projeto foram os seguintes:

- Java *Compiler* (ou Compilador Java): Responsável por compilar o código-fonte dos programas escritos na linguagem Java, gerando as classes responsáveis pela execução das tarefas descritas e que podem ser interpretadas pela máquina virtual do sistema operacional.
- Java *Debugger* (ou Debugador Java): Capacidade de executar uma classe passo-a-passo, verificando possíveis erros em tempo de execução e facilitando o trabalho de tratamento desses erros e/ou falhas.
- *JDBC* (Conectividade com Banco de Dados Java): Oferece um conjunto de tarefas em Java capaz de enviar instruções *SQL* para qualquer banco de dados relacional, nesse caso o MySQL. De maneira geral, serve para manipular dados em um banco de dados.
- *XML*: Arquivo estruturado capaz de conter configurações e atributos para utilização durante a execução dos processos internos às classes.

- *Collections* (Coleções): Lista capaz de conter qualquer objeto, facilitando a manipulação de objetos independente dos atributos e da quantidade dos mesmos.

- *Lang* e *Util*: Pacotes mais utilizados do Java, responsáveis por conter os métodos e funções mais utilizadas em aplicativos Java.

- Java VM *Runtime*: Máquina Virtual que tem por finalidade a interpretação e execução de classes Java previamente compiladas. É a máquina virtual que possibilita que os programas escritos em Java sejam executados em diferentes sistemas operacionais, tornando a linguagem multi-plataformas.

3.9 - **Banco de Dados**

O Banco de Dados escolhido para utilização no desenvolvimento do projeto foi o MySQL devido à sua licença ser gratuita, ao mesmo tempo em que suas funcionalidades são robustas e seu gerenciamento simples. Dentre os bancos de dados Open Source atualmente no mercado, o MySQL é considerado o mais popular, tendo como desenvolvedor, distribuidor e suporte a MySQL AB[9] [**MYSQL - 2006**].

O MySQL foi criado pela SQL AB com o intuito de conectar as tabelas de seus sistemas utilizando rotinas de baixo nível (ISAM[10] [**WIKIPEDIA - ISAM**]). No entanto, após alguns testes, chegou-se à conclusão de que os comandos *SQL* não eram rápidos nem flexíveis o bastante para as necessidades da empresa, resultando assim em uma nova interface *SQL* para o banco de dados, mas mantendo praticamente a mesma *API* do *SQL*. Tal *API* foi escolhida para facilitar a compatibilização dos códigos de terceiros, que eram escritos para uso com o *SQL*, com o uso do MySQL.

[9] MySQL AB é uma empresa que desenvolve e comercializa uma família de produtos de alta performance, servidores de banco de dados de baixo custo e ferramentas.

[10] Padrão para Método de Acesso Seqüencial Indexado, um método para armazenar dados e acessá-los de forma rápida. Originalmente desenvolvido pela IBM, forma atualmente a base

O MySQL é um SGBD Relacional – Sistema de Gerenciamento de Banco de Dados Relacional, ou seja, é utilizado para adicionar, acessar e processar dados armazenados em um banco de dados de um computador. Um banco de dados relacional armazena dados em tabelas separadas, ao invés de colocar todos os dados em um só local. Isso tem como conseqüência velocidade e flexibilidade [**MYSQL - 2006**]. A linguagem utilizada para gerenciar e lidar com os dados do banco é mundialmente conhecida como *SQL – Structured Query Language*, ou Linguagem Estruturada para Consultas. *SQL* apresenta comandos que permitem a definição dos dados, chamada *DDL* (*Data Definition Language*, ou Linguagem de Definição de Dados). Há também comandos da série *DML* (*Data Manipulation Language*, ou Linguagem de Manipulação de Dados), que têm por objetivo a realização de consultas, inserções, alterações ou exclusões de dados nas tabelas. [**PRODEL**]

3.9.1 - Características do MySQL

Dentre as diversas características do MySQL, podem-se destacar as seguintes:

- Interface Visual:

É disponibilizada uma ferramenta para gerenciamento do Banco de Dados chamada MySQL *Administrator*. Trata-se de uma ferramenta visual que serve para a criação e edição de tabelas, bem como gerenciamento dos relacionamentos entre as chaves das tabelas. A Figura 3.9 ilustra a interface do MySQL *Administrator*.

---

para quase todos os tipos de banco de dados. Especificamente no MySQL esse padrão é conhecido como MyISAM.

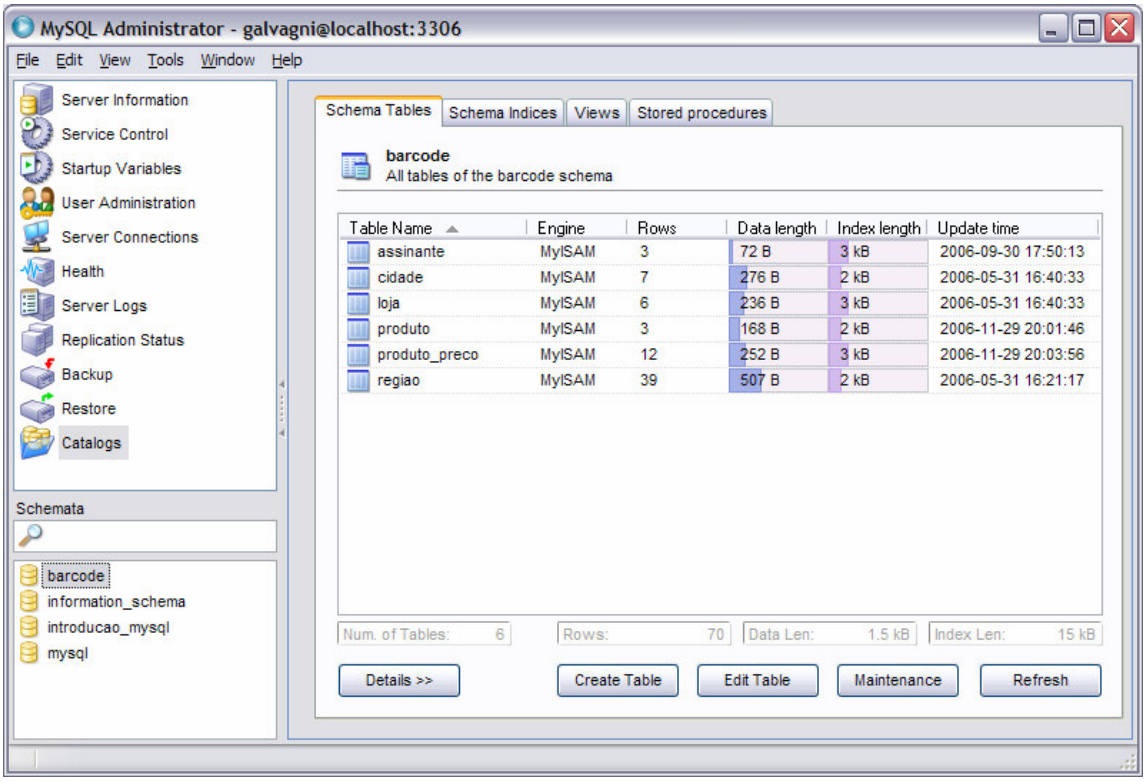

Figura 3.9 – Interface do MySQL *Administrator*

- Manipulação dos Dados:

A manipulação dos dados é realizada através de uma ferramenta específica, o MySQL *Query Browser*. Com essa ferramenta é possível a inserção, deleção e/ou edição dos dados, bem como manter a integridade dos relacionamentos já existentes. A Figura 3.10 ilustra a interface visual da referida ferramenta.

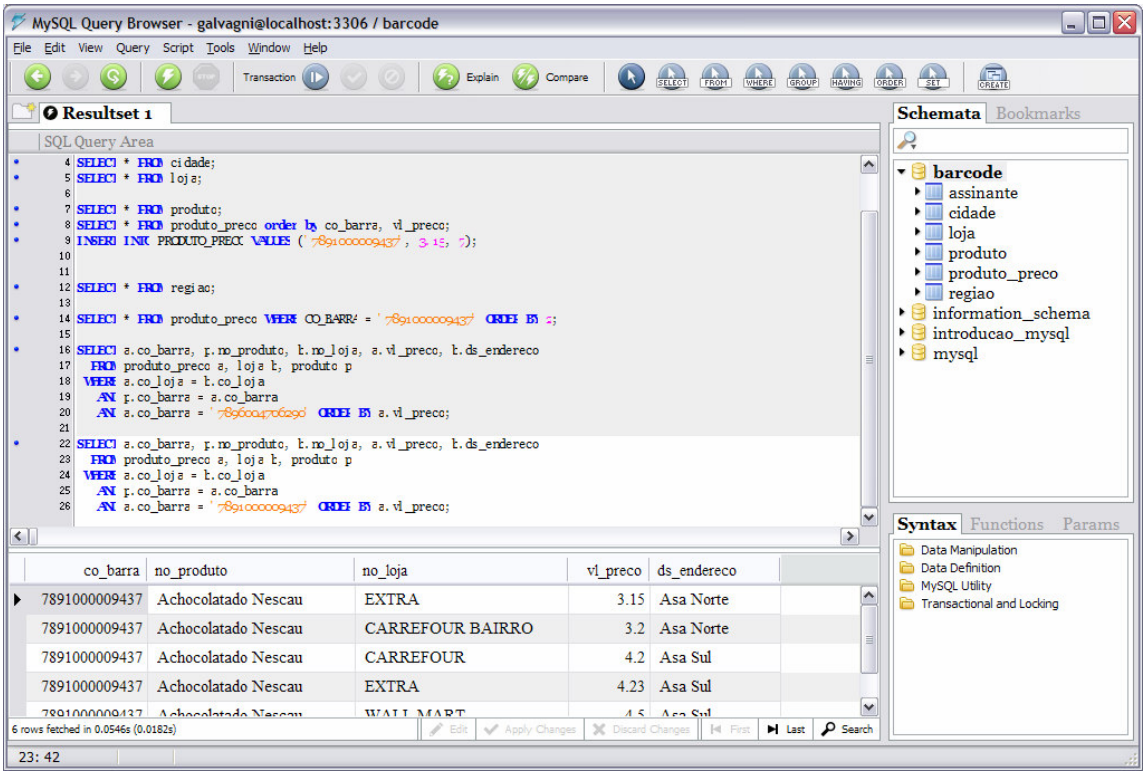

Figura 3.10 – Interface do MySQL *Query Browser*

- Portabilidade:

É escrito em linguagem C e C++, sendo testado por uma ampla faixa de compiladores. Fornece mecanismos de armazenamento transacional e não transacional, bem como tabelas em disco (MyISAM). O acesso a essas tabelas é extremamente rápido pela compressão de índices. Possui um sistema de alocação de memória muito rápido e baseado em processos (*Thread*[11] [**WIKIPEDIA - THREAD**]). Encontra-se disponível como versão cliente/servidor ou embutida (ligada).

- Comandos e Funções:

Completo suporte aos operadores e funções nas consultas, atualizações e/ou exclusões. Suporte pleno às cláusulas *SQL* em geral, bem como para funções de agrupamento

[11] *Thread*, ou linha de execução, é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas simultaneamente. O suporte à thread é fornecido pelo próprio sistema operacional ou implementado através de uma biblioteca de uma determinada linguagem.

(COUNT(), COUNT(DISTINCT), AVG(), STD(), SUM(), MAX() e MIN()). [**MYSQL - 2006**]

- Segurança:

É um sistema com suporte a senhas e privilégios muito flexíveis, seguro e que permite verificações baseadas em estações e máquinas. A segurança no âmbito das senhas se dá através do tráfego das senhas sempre criptografadas na conexão com o banco. [**MYSQL - 2006**]

- Conectividade:

O MySQL permite conexões através de sockets *TCP/IP*[12] [**WIKIPEDIA – TCP/IP**] em qualquer plataforma. A interface *Connector/ODBC* fornece ao MySQL suporte à programas clientes que utilizam conexão *ODBC* (*Open DataBase Conectivity*). Todas as funções *ODBC* são suportadas, assim como muitas outras.

3.9.2 - Suporte e Licença do MySQL

O suporte técnico configura respostas individuais aos problemas ocorridos no ambiente do programa, diretamente dos engenheiros de *software* que codificam o MySQL. Todo e qualquer problema envolvendo o MySQL é de suma importância para os engenheiros de *software*, devido à necessidade de estar sempre aprimorando seus produtos e oferecendo um produto cada vez melhor.

O programa MySQL é distribuído sob a *GNU – General Public License* (*GPL*[13]), que é provavelmente a melhor licença Open Source (Código Aberto) atualmente conhecida

[12] TCP – *Transmission Control Protocol*, ou Protocolo de Controle de Transmissão e o IP – Internet *Protocol*, ou Protocolo de Internet configuram os protocolos para a rede WWW, formando o grupo de protocolos de comunicação que implementama pilha de protocolos sobre a qual a internet e a maioria das redes comerciais funcionam.

[13] Os termos formais da licença *GPL* podem ser encontrados em <http://www.fsf.org/licenses/>.

[**MYSQL - 2006**]. Pela distribuição ser através da *GPL*, sua utilização é geralmente gratuita, porém para certos usos é necessária a aquisição de licenças comerciais da MySQL AB.

Configuram uso da licença *GPL* as seguintes situações:

- Quando o sistema ou aplicação são distribuídos juntamente com o código fonte da MySQL [MYSQL - 2006];
- Quando o código fonte do MySQL é distribuído juntamente com outros programas que não são ligados ou dependentes do sistema do MySQL para suas funcionalidades, mesmo se a distribuição for comercial. Tal forma de licença é conhecida como agregação na licença *GPL* [**MYSQL - 2006**];
- Em uma distribuição de qualquer parte do sistema do MySQL [**MYSQL - 2006**];
- Para provedores de Serviços de Internet (*Internet Service Provider – ISP*) que oferecem hospedagem web com servidores MySQL para seus clientes [**MYSQL - 2006**];
- Na utilização do banco de dados MySQL em conjunto com um servidor web. [**MYSQL - 2006**]

Os dados utilizados durante o desenvolvimento do projeto, bem como nos testes são fictícios e os valores dos produtos meramente ilustrativos.

3.10 - *Softwares* utilizados

Para a simulação do sistema como um todo foram utilizado dois *softwares*, um para o desenvolvimento do *software* de interpretação e da consulta de preços no banco de dados, e outro para a simulação dos celulares que não possuem câmera digital embutida. Ambas as ferramentas são gratuitas e estão disponíveis para download na internet. A seguir, uma visão geral das tecnologias utilizadas.

3.10.1 - Plataforma Eclipse

O projeto Eclipse foi iniciado na IBM que desenvolveu a primeira versão do produto e doou-o como *software* livre para a comunidade. Atualmente, o Eclipse é a IDE Java mais utilizada no mundo. [**WIKIPEDIA - IDE**]

A Comunidade Eclipse está focada no provimento de uma plataforma de desenvolvimento com código aberto e estrutura de aplicações para a construção de *software*s. A Fundação Eclipse é uma fundação sem fins lucrativos formada para avançar na criação, evolução, promoção e suporte à Plataforma Eclipse, cultivando tanto a comunidade de código aberto como um ecossistema de produtos complementares, potencialidades e serviços. [**PLATAFORMA ECLIPSE**]

Trata-se de uma *IDE – Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, ou seja, um programa de computador que serve para o desenvolvimento de outros programas e sistemas [**WIKIPEDIA - IDE**]. A Figura 3.11 ilustra as principais características do Eclipse:

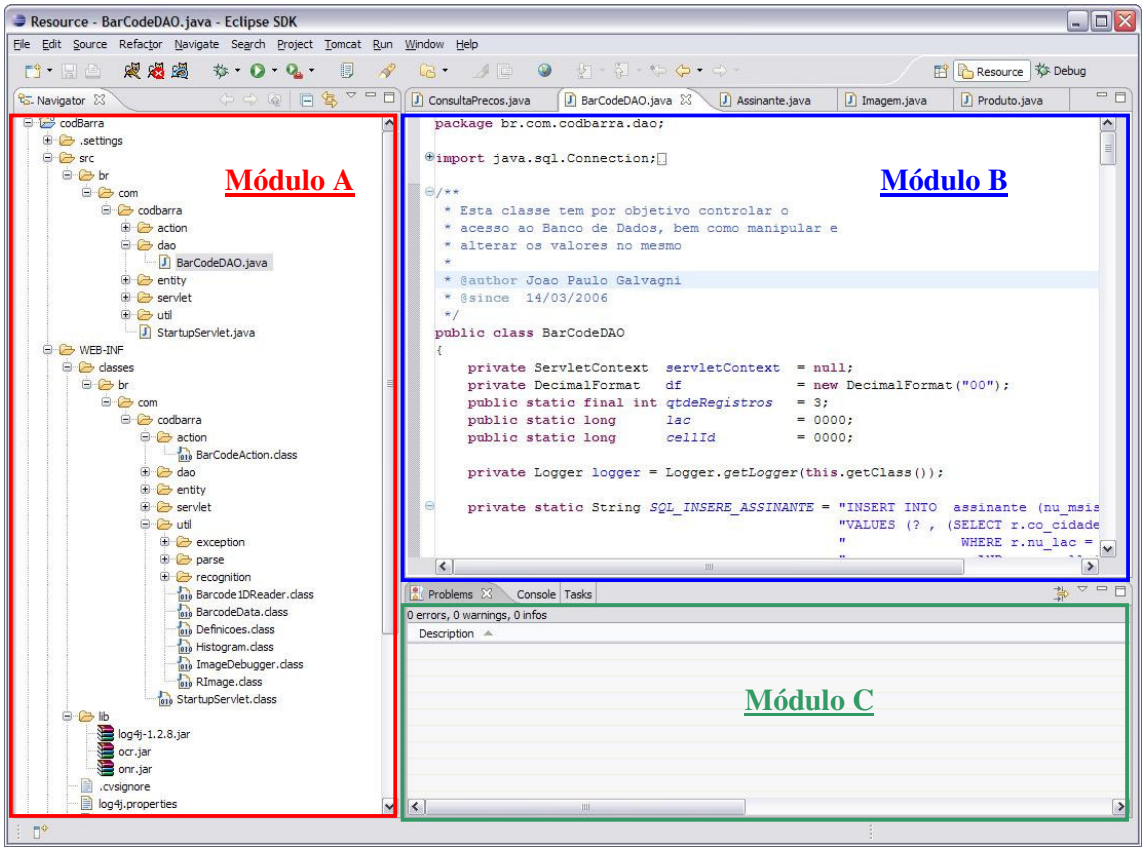

Figura 3.11 – Interface visual do Eclipse

Para entender um pouco da interface visual do Eclipse, seguem algumas características de cada módulo do programa:

- Módulo A – Navegador: Serve para visualizar e administrar as pastas e arquivos que constituem no projeto. Facilita na montagem do projeto, facilitando questões como hierarquia e sub-pastas.

- Módulo B – Editor: Uma dos principais módulos do programa, onde são escritos os códigos que dão origem às classes e rotinas de processamento do sistema.

- Módulo C – Visualizações: Disponibiliza problemas em tempo de desenvolvimento, tarefas a serem executadas, um console para mostrar saídas e resultados, entre vários outros tipos de visualizações.

A utilização da *IDE* Java Eclipse foi fundamental para a conclusão esperada do projeto, pois além de possuir todas as ferramentas necessárias para o desenvolvimento do

sistema como um todo, existem diversas fontes espalhadas pela internet contendo exemplos e modelos de projetos, o que propicia um bom aproveitamento das propriedades do programa.

3.10.2 - WAC

O *WAC* é uma ferramenta de desenvolvimento para a construção de aplicações voltadas para dispositivos móveis, baseados na tecnologia *WIG*. Trata-se de uma ferramenta gratuita, de fácil instalação e útil no desenvolvimento e testes de aplicações *WIG* antes da implantação no ambiente real. [**SMARTTRUST WAC**]

É utilizado para simular um ambiente completo, incluindo a própria Plataforma DP, a rede de telefonia *GSM*, o terminal móvel e o simcard contendo o SmartTrust *WIB* [**SMARTTRUST WAC**]. Abaixo, algumas das principais características do *WAC*:

- Edição de códigos *WIG WML*;
- Verificação e validação de códigos;
- Visualização de códigos no simulador *WIB* do telefone celular;
- Requisições *HTTP* diretamente para aplicações, tendo a capacidade de resolver e simular a resposta, caso essa seja um código *WIG WML.*

Todas essas características estão presentes na versão gratuita do *WAC*, tornando essa uma ferramenta completa e altamente eficiente para o desenvolvimento de aplicações completas para aparelhos celulares.

Na Figura 3.12, ilustra-se a interface do programa com suas principais funções.

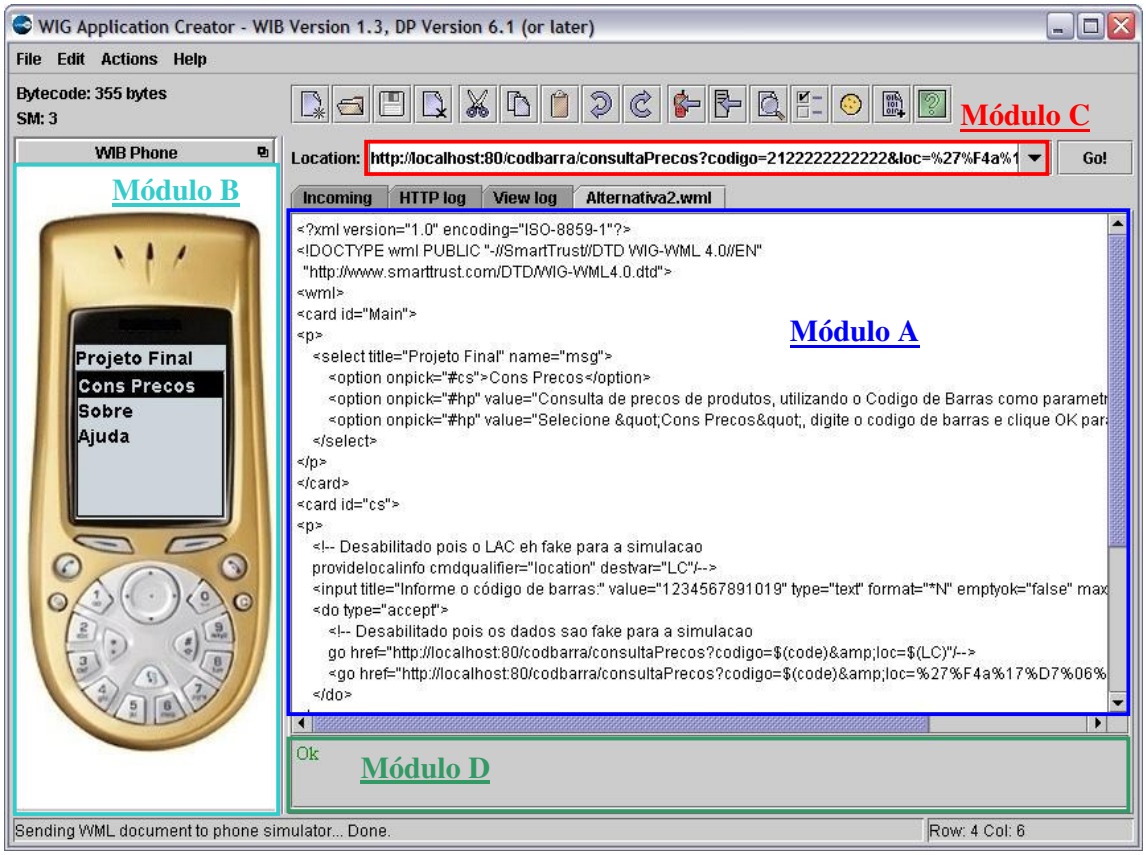

Figura 3.12 – Interface visual do *WAC*

Detalhando os módulos existentes na interface do *WAC* temos:

- Módulo A – Editor de código: Área disponível para a exibição e edição do código-fonte de aplicativos e instruções *WIG WML*;
- Módulo B – Simulador *WIB*: Serve para simular o código-fonte no formato em que será visualizado nos aparelhos celulares com SmartTrust *WIB*;
- Módulo C – Requisições HTTP: Possibilita o envio de requisições via WEB ou localmente, através de URLs;
- Módulo D – Validador de código: Módulo responsável por validar o código-fonte escrito no módulo de edição, acusando o local exato do erro de código.

Com uma interface amigável e funcionalidades robustas, o *WAC* se mostrou uma poderosa ferramenta no auxílio ao desenvolvimento do projeto. Com a devida configuração, a simulação de um fluxo completo de consulta através do *WAC* obteve o sucesso esperado.

Capítulo 4 - Desenvolvimento do Projeto

Neste capítulo serão detalhadas as etapas do reconhecimento do código de barras, mais precisamente a extração do código numérico de uma imagem contendo algum código de barras válido. A idéia é explicar como a tecnologia funciona, bem como sua aplicação real conforme o tema proposto.

O objetivo principal do reconhecimento é, com base em uma fotografia digital de um código de barras tirada a partir de um aparelho celular, extrair a parte numérica do código, considerando exclusivamente as barras, ou seja, a distância entre as barras pretas e brancas – espaços entre as barras pretas, e da espessura das barras pretas, que seguem um padrão matemático. Este e outros processos serão detalhados a seguir.

Será abordado também o *software* de gerenciamento das consultas, onde há o recebimento das requisições e parâmetros de consulta e a elaboração da resposta a ser enviada ao cliente contendo os resultados da comparação.

4.1 - **Banco de Dados Aplicado ao Projeto**

A utilização do banco de dados no projeto se deu pela necessidade de se ter os registros referentes aos produtos – preços e códigos, locais, estabelecimentos comerciais, bem como informações dos assinantes que utilizam o serviço de comparação de preços.

A Figura 4.1 ilutra o modelo físico das tabelas e seus respectivos relacionamentos:

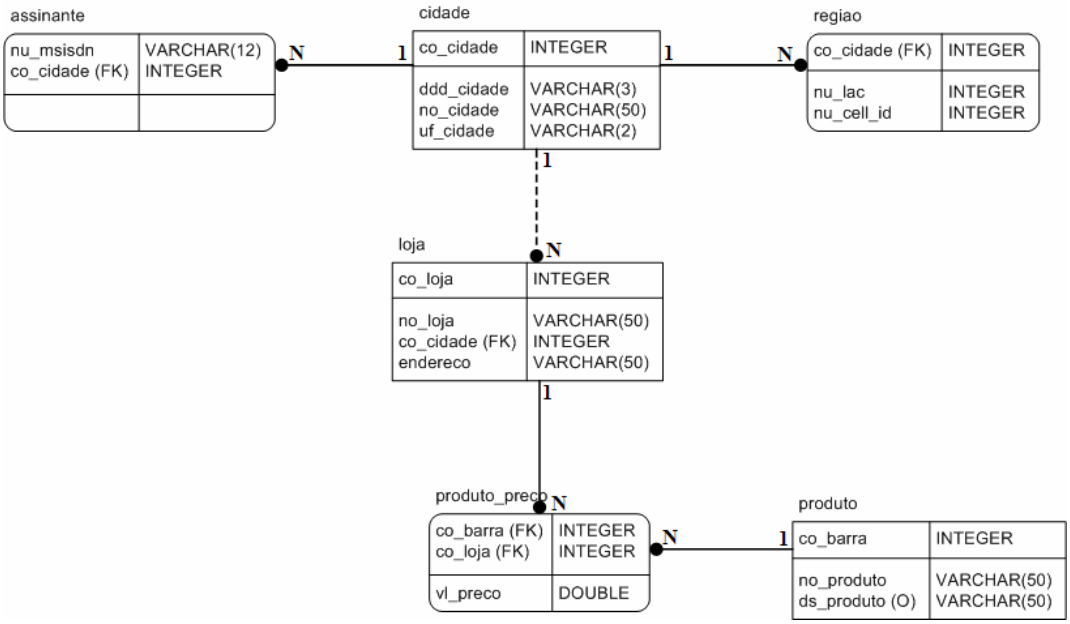

Figura 4.1 – Modelo Relacional do Banco de Dados

4.1.1 - Detalhamento das tabelas

- Tabela ***assinante***:

Esta tabela contém as informações dos clientes, ou seja, seu MSISDN[14] [**WIKIPEDIA - MSISDN**] e o código da cidade a qual pertence. Dessa forma, sabe-se em qual cidade e região a consulta será efetuada, mantendo a proximidade entre o cliente e o estabelecimento que contem o produto em questão. Campos da tabela: *nu_msisdn*: Número do cliente; *co_cidade*: Código da cidade em que o assinante se encontra.

- Tabela ***cidade***:

Contém as informações das cidades cadastradas no sistema, tais como o código da cidade, o DDD (Código Nacional) da mesma, seu nome e sua UF (Unidade Federativa), ou seja, a sigla do estado o qual pertence a cidade. Campos da tabela: *co_cidade*: Código da

[14] MSISDN (ou Serviço Digital Integrado à Rede de Estações Móveis: Para o projeto, são considerados os 12 dígitos referentes ao número completo do cliente, determinado pelo código do país (no caso do Brasil o código é o 55), seguido do Código Nacional sem o dígito "0" (Ex.: 61 – Brasília, 62 – Goiânia, etc.) e por último o número da linha telefônica do cliente.

cidade; *ddd_cidade*: Código Nacional da cidade; *no_cidade*: Nome da cidade; *uf_cidade*: Unidade Federativa, mais conhecida como código das cidades.

- Tabela ***regiao***:

Possui como chave primária o código da cidade, chave essa controlada pela tabela ***cidade***. Há também os campos nu_lac e cell_id, que juntos determinam a antena utilizada no momento da realização da consulta, permitindo assim uma maior precisão com relação ao retorno da consulta e comparação dos preços. Campos da tabela: *co_cidade*: Código da cidade; *nu_lac* e *cell_id*: juntos configuram a antena da empresa de telecomunicação.

- Tabela ***loja***:

Integração entre as tabelas ***cidade*** e ***produto_preco***, devido à necessidade de se determinar a que cidade a loja pertence e a que loja o produto com um determinado preço pertence. Campos da tabela: *co_loja*: Código da loja; *no_loja*: Nome da loja; *co_cidade*: Código da cidade que a loja pertence; *endereco*: Detalhamento ou referência da loja.

- Tabela ***produto_preco***:

Contém as informações dos produtos e das lojas que o possuem, juntamente com o preço de cada produto. Campos da tabela: *co_barra*: Código de barras do produto; *co_loja*: Código da loja; *vl_preco*: Valor do produto para cada loja.

- Tabela ***produto***:

Possui os detalhes dos produtos comercializados, bem como o código de barras que é único por produto. Campos da tabela: *co_barra*: Código de barras do produto; *no_produto*: Nome comercial do produto; *ds_produto*: Breve descrição do produto.

4.2 - **Reconhecimento do Código de Barra**

O reconhecimento do código de barras está associado a uma série de etapas seqüenciais, cada qual com sua importância e nível de complexidade. A seguir será detalhada cada etapa do processo, desde o escaneamento da imagem até a validação do dígito verificador do código de barras em seu formato numérico.

- Leitura da imagem e Carregamento em memória

Para o devido tratamento da imagem, é necessária a leitura da mesma, ou seja, carregar a imagem em memória, transformando a mesma em valores binários, tornando possível o devido reconhecimento computacional da mesma. Essa imagem em memória já conterá informações como altura, largura e tamanho.

Considerando a impossibilidade de utilizar o ambiente real de uma empresa de telecomunicações, a alternativa encontrada foi a leitura da imagem previamente salva em um diretório local. Para simular o processo de requisição, será enviado um comando através do *software WIG Application Creator* (*WAC*) para que o programa responsável pelo controle das requisições possa ser iniciado.

- Propriedades da imagem

A imagem passará por um processo de aplicação de contraste e pelo processo de transformação no padrão Preto e Branco, ou seja, deixar a imagem mais clara e homogênea, facilitando o processamento da mesma. Como padrão, todas as imagens irão passar pelo processo de transformação em Preto e Branco.

- Escaneamento da imagem

A imagem, ou melhor, os dados da imagem serão colocados em vetores, linha a linha, facilitando a manipulação da mesma. Dessa maneira, a imagem é considerada com apenas uma dimensão, ou seja, não há vetores maiores do que 1x1, mas sim um conjunto de

vetores NxN, mantendo o processamento de forma unitária, ou seja, as linhas serão tratadas uma por vez.

A etapa seguinte é achar, inserido nos dados da imagem, as “barras pretas” e “espaços em branco” e, consequentemente o valor numérico dos intervalos. Depois de encontrado o código numérico e extraí-lo, o mesmo deverá ser validado, ou seja, será realizado um cálculo para validar o dígito verificador.

Após todo o processo de reconhecimento do código de barras, o processo que se segue é o da consulta dos menores preços no banco de dados, detalhado no tópico seguinte.

4.3 - *Software* de Gerenciamento das Consultas

Para a consulta e comparação entre os preços, foram criadas algumas classes que farão esse controle, considerado de entrada e saída, uma vez que haverá o recebimento de dados e o envio dos resultados ao cliente.

A consulta consiste em receber o código de barras em seu formato numérico, já com o dígito verificador válido e acessar o banco de dados para a realização da consulta. Outra informação que é levada em consideração na consulta é a localização do cliente, ou a antena mais próxima do aparelho celular, otimizando os resultados. A idéia é que ao saber os locais onde os preços são menores o cliente possa se deslocar ao estabelecimento informado para realizar a compra do produto. O fluxo do sistema pode ser visualizado na Figura 4.2.

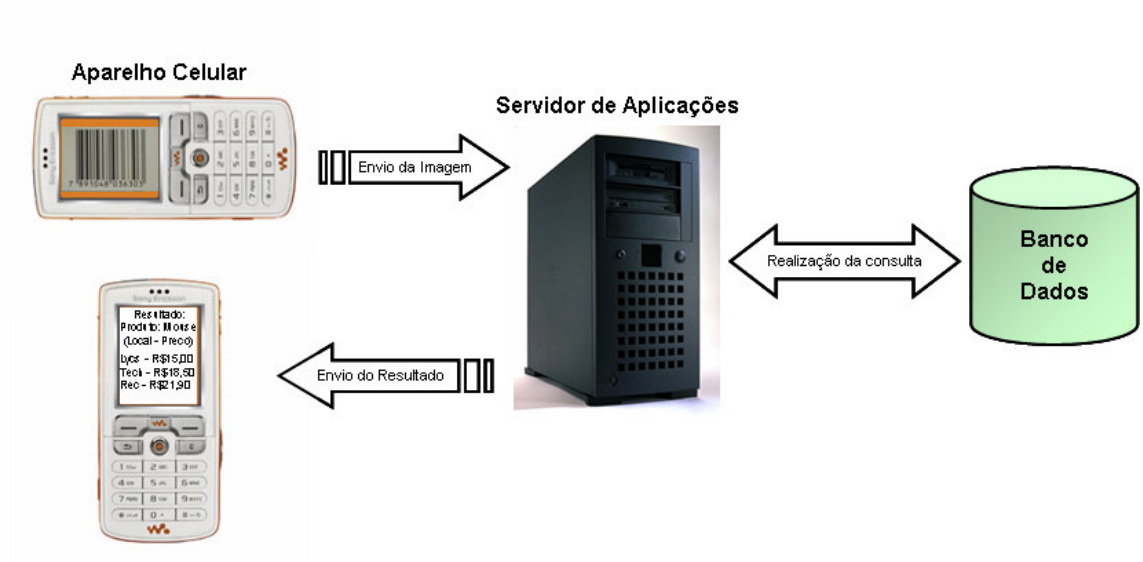

Figura 4.2 – Fluxo Básico do Sistema

O fluxo se inicia com o cliente do serviço tirando uma fotografia digital do código de barras de um produto qualquer. Em seguida o cliente deve enviar a imagem ao servidor de aplicações para que haja o tratamento da imagem, a extração do código de barras no formato numérico e a validação do código, através do cálculo do dígito verificador.

Uma vez que a aplicação possui o número representativo do código, este é enviado ao gerenciador de consultas para então realizar o acesso ao banco e selecionar os produtos com menores preços relacionados àquele código. Tudo isso levando em consideração a cidade e a região em que o cliente está, possibilitando ao cliente se dirigir às lojas mais próximas possíveis. O cliente, por sua vez, também deverá estar cadastrado no banco de dados, caso contrário este será cadastrado no momento anterior à consulta, já com seus dados regionais.

Após a consulta, a aplicação então monta a mensagem de retorno baseado na quantidade de preços definida na configuração do sistema e envia a resposta ao cliente. Visualizado o resultado, o cliente então pode ter uma idéia de quanto poderia economizar ou está economizando naquele momento. Nesse momento o fluxo se completa e o propósito do trabalho alcançado. O tempo para o fluxo completo pode variar, pois depende de condições da rede de telefonia, de plataformas de serviços e servidores de aplicações.

4.4 - **Alternativa para Aparelhos sem Câmera Fotográfica**

Aparelhos celulares com câmera digital embutida estão cada dia mais comum, porém o preço ainda não agrada a todos. Sendo assim, como solução alternativa para esse problema, foi criada uma forma de consulta e comparação utilizando um menu interativo já presente nos simcards.

Um exemplo do fluxo a ser seguido por um celular mais simples pode ser visualizado na Figura 4.3.

Figura 4.3 – Fluxo alternativo para aparelhos simples

O procedimento é bem simples e com alguns cliques o cliente é capaz de obter o resultado esperado. Ao acessar o Menu do aparelho celular, o cliente receberá algumas opções de serviços e ofertas promocionais. Com a proposta de oferecer uma ferramenta para consulta e comparação de preços, o menu para essa opção se encontraria na opção “Serviços”, em um sub-menu com uma opção “Compara Preços". Ao selecionar essa última, o cliente irá digitar o código de barras em seu formato numérico, também presente nas embalagens dos produtos, e enviar a requisição para o servidor de aplicações. Após a devida comparação de preços, o cliente então recebe como resposta uma mensagem contendo os menores preços e o local de cada um.

O processo de consulta e comparação é semelhante ao descrito no tópico anterior, exceto pelo fato de que o código de barras já é informado ao *software* de consulta em seu

formato numérico, não necessitando processar qualquer imagem. De certa forma, isso torna o serviço mais rápido, ainda que o cliente tenha que digitar o código de barras no celular.

Com essa alternativa, a solução beneficiará não apenas pessoas com um bom poder aquisitivo, mas sim boa parte dos 95 milhões de clientes ativos existentes no Brasil atualmente (fonte: Anatel). [**ANATEL**]

Capítulo 5 - Conclusão

Este trabalho apresenta a simulação de uma ferramenta pioneira para os usuários da telefonia móvel celular, que é a consulta e comparação de preços de produtos comerciais. Foram utilizadas tecnologias já existentes no mercado, mas que em conjunto formam um sistema completo capaz de trazer economia e satisfação para os clientes, bem como maior competitividade e um aumento na concorrência entre os estabelecimentos comerciais.

O reconhecimento de código de barras é uma tecnologia universalmente conhecida e aplicada. Contudo, a tecnologia utilizada para leitura e interpretação dos códigos é a de leitores ópticos, ou seja, feixes de laser, softwares e hardwares específicos. Para a implementação do presente trabalho foi desenvolvido um software exclusivo, de modo a oferecer uma ferramenta rápida, eficiente e de baixo custo.

O desenvolvimento dessa ferramenta exigiu um estudo aprofundado da linguagem de programação Java e dos serviços já existentes nas empresas de telecomunicações. A principal dificuldade encontrada foi o desenvolvimento do software de reconhecimento de código de barras, que é o núcleo do sistema de comparação de preços proposto neste projeto.

A solução final do presente trabalho visa atender a todos os usuários de telefonia móvel celular com a tecnologia GSM, podendo ser utilizada tanto por aqueles que possuem aparelhos celulares de ponta, com câmeras digitais de alta qualidade, quanto por aqueles que possuem os mais simples aparelhos celulares, bastando para tanto um menu interativo disponibilizado pelas empresas de telefonia móvel.

A tecnologia de códigos de barra está em constante desenvolvimento no mercado mundial, existindo uma tendência natural de que novas tecnologias relacionadas ao seu uso continuem sendo implementadas. Levando isso em consideração, alguns trabalhos relacionados ao presente são possíveis, tais como:

- Sobre a interpretação de código de barras, uma solução alternativa poderia ser criada para englobar os padrões de código diferente do EAN.UCC-13.
- Ainda sobre a interpretação de códigos, há a possibilidade de se desenvolver um aplicativo utilizando a tecnologia Java 2 Micro Edition (J2ME) para rodar em celulares com suporte a Java. Dessa forma, a interpretação e tratamento da imagem seriam realizados ainda no aparelho celular.
- Utilizar a idéia de consulta e comparação utilizando tecnologias alternativas, como chips emissores de rádio freqüência, por exemplo.

Referências Bibliográficas

[**ANATEL**] – Anatel. Disponível em: <http://www.anatel.gov.br/Tools/frame.asp?link=/comunicacao_movel/smc/smc_smp_dados_por_uf.pdf >. Acessado em: 10/11/2006.

[EAN BRASIL – 2006] – EAN Brasil. Disponível em: <http://www.eanbrasil.org.br/servlet/ServletHomePage?lang=pt_BR >. Acessado em: 20 fev. 06.

[EAN BRASIL – EAN.UCC] – EAN Brasil. Disponível em: <http://www.eanbrasil.org.br/servlet/ServletContent?requestId=10 >. Acessado em: 20 fev. 06.

[**EAN BRASIL – Sistema de Numeração**] – EAN Brasil. Disponível em: <http://www.eanbrasil.org.br/servlet/ServletContent?requestId=21 >. Acessado em: 21 fev. 06.

[GS1 BRASIL] – GS1 Brasil. Disponível em: <http://www.eanbrasil.org.br/servlet/ServletContent?requestId=20&id:news=3301&id:newsclassification=1&lang=pt_BR >. Acessado em: 21 fev. 06.

[GS1 U.S.] – GS1 U.S., Global Standard United States. Disponível em: <http://www.gs1us.org/history.html >. Acessado em: 17 fev. 06.

[ITUDOMAIS] – Itudomais. Disponível em: <http://www.itudomais.com.br/dici/dicionario.htm > . Acessado em: 29 mar. 06.

[**MYSQL - 2006**] – MySQL. Disponível em: <http://www.mysql.com >. Acessado em: 10 ago. 06.

[**PLATAFORMA ECLIPSE**] – Plataforma Eclipse. Disponível em: <http://www.eclipse.org/org >. Acessado em: 09 nov. 06.

[**PRODEL**] – Prodel. Disponível em: <http://www.prodel.com.br/sql.htm >. Acessado em: 12 ago. 06.

[**SMARTTRUST WAC**] – SmartTrust WAC. Disponível em: <http://www.smarttrust.com/support_services/services_wac.asp >. Acessado em: 08 nov. 06.

[**WENZEL, FLAVIO**] – Informatiquês. Disponível em: <http://www.flaviowenzel.hpg.ig.com.br/informatiques/o.html >. Acessado em: 20 out. 06.

[**WIKIPEDIA - *API***] – *API*. Disponível em: <http://pt.wikipedia.org/wiki/*API* >. Acessado em: 08 nov. 06.

[**WIKIPEDIA - CHAT**] – Chat. Disponível em: <http://pt.wikipedia.org/wiki/Chat >. Acessado em: 12 out. 06.

[**WIKIPEDIA - *HTML***] – *HTML*. Disponível em: <http://pt.wikipedia.org/wiki/*HTML* >. Acessado em: 10 set. 06.

[**WIKIPEDIA - HTTP**] – HTTP. Disponível em: <http://pt.wikipedia.org/wiki/HTTP >. Acessado em: 10 out. 06.

[**WIKIPEDIA - IDE**] – IDE. Disponível em: <http://pt.wikipedia.org/wiki/Ambiente_de_Desenvolvimento_Integrado >. Acesso em 08 nov. 06.

[**WIKIPEDIA - ISAM**] – ISAM. Disponível em: <http://en.wikipedia.org/wiki/ISAM >. Acessado em: 12 ago. 06

[**WIKIPEDIA - MSISDN**] – MSISDN. Disponível em: <http://en.wikipedia.org/wiki/MSISDN >. Acessado em: 15 ago. 06.

[**WIKIPEDIA - RINGTONES**] – Ringtones. Disponível em: <http://en.wikipedia.org/wiki/Ringtones >. Acessado em: 12 out. 06.

[**WIKIPEDIA - *SMS***] – *Short Message Service*. Disponível em: <http://en.wikipedia.org/wiki/Short_message_service >. Acessado em: 30 ago. 06.

[**WIKIPEDIA - *TCP/IP***] – *TCP/IP*. Disponível em: <http://pt.wikipedia.org/wiki/*TCP/IP* >. Acessado em: 12 ago. 06.

[**WIKIPEDIA - THREAD**] – *Thread*. Disponível em: <http://pt.wikipedia.org/wiki/*Thread* >. Acessado em: 12. ago. 06.

[**WIKIPEDIA – VIRTUAL MACHINE**] – Virtual Machine. Disponível em: <http://pt.wikipedia.org/wiki/Virtual_Machine >. Acessado em: 12 out. 06.

[**WIKIPEDIA - WAP PUSH**] – WAP PUSH. Disponível em: <http://pt.wikipedia.org/wiki/ WAP >. Acessado em: 30 ago. 06.

Apêndice CODIGOS-FONTE DO PROJETO

A seguir serão detalhados os códigos-fonte utilizados no projeto e suas principais características e funções.

- **PriceComparison.java**

Servlet criada para gerenciar as requisições de consulta e comparação de preços, redirecionando as requisições para mostrar a resposta aos cliente.

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Collection;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;
import org.apache.log4j.Logger;
import br.com.codbarra.dao.BarCodeDAO;
import br.com.codbarra.util.Definitions;
import br.com.codbarra.util.Validator;

public class PriceComparison extends HttpServlet
{
	private Context initContext  = null;
	private String  server = null;
	private Logger logger = Logger.getLogger(this.getClass());

	public void init(ServletConfig arg0) throws ServletException
	{
		try
		{
			initContext    = new InitialContext();
			server         = (String)arg0.getServletContext().getAttribute("Servidor");
		}
		catch (NamingException e)
		{
			logger.error(e);
		}
	}

	protected void service(HttpServletRequest request, HttpServletResponse response)
	throws ServletException, IOException
	{
		Connection con = null;
		PrintWriter out = response.getWriter();

		try
```

```
		{
			// Recebe os seguintes parametros:
			// MSISDN  - Numero do assinante
			// loc      - Codigo de localizacao do assinante
			// barcode - Codigo de barras com 13 digitos

			String msisdn = request.getParameter("MSISDN");
			String loc        = request.getParameter("loc");
			String code = request.getParameter("codigo");

			logger.debug("Parametros recebidos: MSISDN: " + msisdn + "\nLOC: " + loc +
"\nCODIGO: " + code);

			// Cria uma nova instancia do BarCodeDAO e do objeto BarCode
			BarCodeDAO barCodeDAO = new BarCodeDAO();
			barCodeDAO.setERB(loc);

			if (Validator.validateBarCode(code))
			{
				logger.debug("Codigo de barras validado com sucesso!");
				con = getConnection();

				if (barCodeDAO.getAssinante(con, msisdn) == null)
				{
					logger.debug("Assinante " + msisdn + " nao existe.");

					// Insere o assinante no BD
					barCodeDAO.insereAssinante(con, msisdn);
				}

				// Colecao contendo os objetos BarCode
				Collection listaDePrecos = barCodeDAO.findPrecos(con, msisdn, code);

				// Setando o atributo contendo a lista dos objetos BarCode
				request.setAttribute("listaDePrecos", listaDePrecos);

				logger.debug("Encaminhando a resposta para o cliente.");

				// Redireciona a requisicao para mostrar o resultado
				request.getRequestDispatcher("/ShowPrices.jsp").forward(request,response);
			}
			else
			{
				request.setAttribute("message", "Erro na validacao do codigo de barras.");
				request.getRequestDispatcher("/ShowMessage.jsp").forward(request, response);
			}
		}
		catch (Exception e)
		{
			logger.error("ERRO: ", e);
			// Mostra uma mensagem de erro ao cliente e a opcao de uma nova tentativa
			out.println(Definitions.CABECALHO_WML + "Erro inesperado. Tente novamente em
alguns instantes.\n" + Definitions.RODAPE_WML);
		}
		finally
		{
			out.flush();
			out.close();
			try
			{
```

```
                        if (con != null && !con.isClosed())
                                con.close();
                        }
                        catch (SQLException e)
                        {
                                logger.error("Erro no fechamento da conexao. ERRO: ", e);
                        }
                }
        }

        public Connection getConnection() throws SQLException, NamingException
        {
                DataSource ds = null;
                Connection connection = null;
                try
                {
                        ds = (DataSource) initContext.lookup("java:comp/env/jdbc/MySQLDB");
                        connection = ds.getConnection();
                }
                catch(SQLException e)
                {
                        logger.error("Erro na tentativa de conexao com o Banco de Dados.", e);
                }
                logger.debug("Conexao efetuada com sucesso.");
                return connection;
        }
}
```

- **ReadBarCode.java**

Servlet utilizada para simular o recebimento de uma imagem e gerenciar a extração do código de barras numérico da mesma. Após o processamento da imagem, esta classe redireciona a requisição para a Servlet PriceComparison.java, que por sua vez irá realizar o mesmo processo de consulta e retorno dos resultados ao cliente.

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.log4j.Logger;
import br.com.codbarra.action.ReadBarCodeAction;
import br.com.codbarra.entity.BarCodeImage;
import br.com.codbarra.util.Definitions;

public class ReadBarCode extends HttpServlet
{
        private Context initContext = null;
        private String  server          = null;
        private Logger logger                   = Logger.getLogger(this.getClass());
```

```
	public void init(ServletConfig arg0) throws ServletException
	{
		try
		{
			initContext	= new InitialContext();
			server		= (String)arg0.getServletContext().getAttribute("Servidor");
		}
		catch (NamingException e)
		{
			logger.error(e);
		}
	}

	protected void service(HttpServletRequest request, HttpServletResponse response)
	throws ServletException, IOException
	{
		// SERVE PARA IMPRIMIR OS TESTES APENAS
		PrintWriter out = response.getWriter();

		// Recebe os seguintes parametros:
		// MSISDN   - Numero do assinante
		// fileName - Nome do arquivo a ser lido
		String msisdn		= request.getParameter("MSISDN");
		String fileName		= request.getParameter("fileName");
		String loc		= request.getParameter("loc");

		try
		{
			logger.debug("Parametros recebidos: MSISDN: " + msisdn + "\nNome do Arquivo: " +
fileName);
			ReadBarCodeAction readBarCode = new ReadBarCodeAction();
			BarCodeImage barCodeImage = readBarCode.scanImage(fileName, Definitions.EAN13);

			if ( !Definitions.COD_ERRO_STR.equals(barCodeImage.getCodeStr()) )
			{
				request.setAttribute("MSISDN", msisdn);
				request.setAttribute("codigo", barCodeImage.getCodeStr());
				request.setAttribute("loc", loc);

				// Redireciona a requisicao para mostrar o resultado
				request.getRequestDispatcher("/priceComparison").forward(request,response);
			}
			else
			{
				request.setAttribute("message", barCodeImage.getDescription());

				// Redireciona a requisicao para mostrar uma mensagem ao assinante
				request.getRequestDispatcher("/ShowMessage").forward(request, response);
			}
		}
		catch (Exception e)
		{
			logger.error("ERRO: ", e);
			// Mostra uma mensagem de erro ao cliente e a opcao de uma nova tentativa
			out.println(Definitions.CABECALHO_WML + "Erro inesperado. Tente novamente em
alguns instantes.\n" + Definitions.RODAPE_WML);
		}
		finally
		{
```

```
                    out.flush();
                    out.close();
                }
        }
}
```

- **Classe Definitions.java**

Classe responsável por conter os parâmetros padrões utilizados no projeto, parâmetros esses que não sofrem alterações, ou seja, s

```
public class Definitions
{
        public static final String DIR = "c:\\temp\\";
        public static final String CABECALHO_WML = "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"
?>\n"
                                "<!DOCTYPE wml PUBLIC \"-//SmartTrust//DTD WIG-WML 4.0//EN\"\n" +
                                "http://www.smarttrust.com/DTD/WIG-WML4.0.dtd\">\n" +
                                "<wml>\n" +
                                "<card>\n" +
                                "<p>\n";

        public static final String RODAPE_WML = "</p>\n" +
                                                "</card>\n" +
                                                "</wml>" ;

        public static final boolean DEBUG_ON  = true;
        public static final boolean DEBUG_OFF = false;

        public static int FOREGROUND = 0;
        public static int BACKGROUND = 1;

        public static String COD_ERRO_STR = "-1";
        public static int     COD_ERRO_INT = -1;

        public static final int qtdeRegistros     = 3;

        public static final int EAN13             = 4;
        public static final int CODE128           = 1;
        public static final int IDENTCODE     = 128;
}
```

- **ReadBarCodeAction**

Classe que possui os métodos para leitura e carregamento da imagem em memória a partir do nome do arquivo informado e o método para escanear a imagem, extraindo assim a parte numérica da mesma.

```
import java.awt.Canvas;
import java.awt.Color;
import java.awt.Graphics;
```

```
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.io.FileInputStream;
import org.apache.log4j.Logger;
import br.com.codbarra.entity.BarCodeImage;
import br.com.codbarra.entity.RImage;
import br.com.codbarra.util.Barcode1DReader;
import br.com.codbarra.entity.BarcodeData;
import br.com.codbarra.util.Definitions;

public class ReadBarCodeAction
{
	private Logger logger = Logger.getLogger(this.getClass());

	public BarCodeImage scanImage(String fileName, int type) throws Exception
	{
		BarCodeImage result = new BarCodeImage();
		result.setImage(loadImageFromFile(Definitions.DIR + fileName));

		Barcode1DReader reader = new Barcode1DReader();
		reader.setSymbologies(type);

		RImage rImage = new RImage((BufferedImage) result.getImage());
	long tim = System.currentTimeMillis();

	// Inicio da interpretacao...
	logger.info("Inicio da interpretacao do codigo: " + tim + ". ");
	System.out.println("Inicio da interpretacao do codigo: " + tim + ". ");

	BarcodeData[] barcodes = reader.scan(rImage);
		rImage = null;
		result.setImage(null);

		// Final da interpretacao...
		logger.info("Fim da interpretacao: " + (System.currentTimeMillis() - tim) + ".");
		System.out.print("Fim da interpretacao: " + (System.currentTimeMillis() - tim) + ".");

		if (barcodes.length == 0)
		{
			result.setDescription("*** CODIGO DE BARRAS NAO ENCONTRADO ***");
			result.setCode(Definitions.COD_ERRO_INT);
			result.setCodeStr(Definitions.COD_ERRO_STR);
		}

		for (int i = 0; i < barcodes.length; i++)
		{
			result.setDescription("Codigo de barras encontrado com sucesso.");
			result.setCode(Integer.parseInt(barcodes[i].toString()));
			result.setCodeStr(barcodes[i].toString());
		}

		logger.info(result.getDescription());
		System.out.println(result.getDescription());
		return result;
	}

	private Image loadImageFromFile(String file) throws Exception
	{
```

```
            Image im2 = null;
            MediaTracker mt2 = null;

            FileInputStream in = null;
            byte[] b = null;
            int size = 0;

            in = new FileInputStream(file);

            if (in != null)
            {
                  size = in.available();
                  b = new byte[size];
                  in.read(b);
                  im2 = Toolkit.getDefaultToolkit().createImage(b);
                  in.close();
            }

            mt2 = new MediaTracker(new Canvas());
            if (im2!=null)
            {
                  if (mt2!=null)
                  {
                        mt2.addImage(im2,0);
                        mt2.waitForID(0);
                  }
            }

            BufferedImage input = new BufferedImage(im2.getWidth(null), im2.getHeight(null),
BufferedImage.TYPE_INT_ARGB);

            Graphics g = input.createGraphics();
            g.setColor(Color.white);
            g.fillRect(0,0,im2.getWidth(null),im2.getHeight(null));
            g.drawImage(im2,0,0,null);
            g.dispose();
            g = null;

            return input;
      }
}
```

- **BarCodeDAO**

Classe com características de um DAO (Data Access Object), ou seja, serve como interface entre as aplicações e o banco de dados, através do acesso aos dados inseridos no mesmo.

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Collection;
import javax.servlet.ServletContext;
import org.apache.log4j.Logger;
```

```
import br.com.codbarra.entity.Product;
import br.com.codbarra.entity.Client;
import br.com.codbarra.util.Definitions;

public class BarCodeDAO
{
	private ServletContext servletContext = null;
	private DecimalFormat df = new DecimalFormat("00");
	public static long lac = 0000;
	public static long cellId = 0000;

	private Logger logger = Logger.getLogger(this.getClass());

	private static String SQL_INSERE_ASSINANTE = "INSERT INTO  assinante (nu_msisdn, co_cidade) "
+
	" VALUES (? , (SELECT r.co_cidade FROM regiao r " +
	" WHERE r.nu_lac = ? AND r.nu_cell_id = ?)) " ;

	private static String SQL_CONSULTA_ASSINANTE = "SELECT a.co_cidade, b.no_cidade"  +
	"  FROM assinante a, cidade b"        +
	" WHERE a.nu_msisdn = ? AND a.co_cidade = b.co_cidade" ;

	private static String SQL_ATUALIZA_ASSINANTE = "UPDATE assinante " +
	" SET (co_cidade = (SELECT r.co_cidade FROM regiao r" +
	" WHERE r.nu_lac = ? AND r.nu_cell_id = ?))" +
	" WHERE nu_msisdn = ?";

	private static String SQL_CONSULTA_PRECOS = "SELECT b.no_produto, c.no_loja, d.vl_preco" +
	"  FROM assinante a, produto b, loja c, produto_preco d" +
	" WHERE a.co_cidade = c.co_cidade AND c.co_loja = d.co_loja" +
	"   AND b.co_barra = d.co_barra AND a.nu_msisdn = ?" +
	"   AND b.co_barra = ? ORDER BY d.vl_preco" ;

	public Client getAssinante (Connection con, String msisdn) throws SQLException
	{
		PreparedStatement pstmt = null;
		ResultSet rs = null;
		Client assinante = null;

		try
		{
			pstmt = con.prepareStatement(SQL_CONSULTA_ASSINANTE);

			// Seta o MSISDN para a consulta do Assinante
			pstmt.setString(1, msisdn);

			// executa a consulta
			rs = pstmt.executeQuery();

			// Caso o assinante exista, o resultado se torna verdadeiro
			if (rs.next())
			{
				// Nova instancia do objeto Assinante
				assinante = new Client();

				// Popula o objeto assinante com os valores da consulta
				assinante.setMsisdn(msisdn);
				assinante.setCoCidade(rs.getInt("co_cidade"));
				assinante.setNoCidade(rs.getString("no_cidade"));
			}
```

```
        }
        catch (SQLException e)
        {
                logger.error("Erro na selecao do assinante. ERRO: ", e);
                throw e;
        }
        finally
        {
                if (rs != null)
                        rs.close();
        }
        // Retorna o objeto assinante
        return assinante;
}

public void insereAssinante (Connection con, String msisdn) throws SQLException
{
        PreparedStatement pstmt     = null;

        try
        {
                // Prepara a chamada a insercao do assinante
                pstmt = con.prepareStatement(SQL_INSERE_ASSINANTE);

                // Seta os valores para atualizacao do Assinante no Banco de Dados
                pstmt.setString(1,msisdn);
                pstmt.setLong(2,lac);
                pstmt.setLong(3,cellId);

                // Executa a insercao no BD
                pstmt.executeUpdate();
        }
        catch (SQLException e)
        {
                logger.error("Erro na tentativa de inserir o assinante no BD. ERRO: ", e);
                throw e;
        }
}

public void atualizaAssinante (Connection con, String msisdn) throws SQLException
{
        PreparedStatement    pstmt   = null;
        ResultSet rs = null;

        try
        {
                pstmt = con.prepareStatement(SQL_ATUALIZA_ASSINANTE);

                // Seta os valores necessarios para a atualizacao
                pstmt.setLong(1, lac);
                pstmt.setLong(2, cellId);
                pstmt.setString(3, msisdn);

                // Executa a atualizacao do assinante
                pstmt.executeUpdate();
        }
        catch (SQLException e)
        {
                logger.error("Erro na tentativa de atualizar o assinante. ERRO: " + e);
                con.rollback();
```

```
                throw e;
        }
        finally
        {
                if (rs != null)
                        rs.close();
        }
}

public Collection findPrecos(Connection con, String msisdn, String codigo) throws SQLException
{
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        Collection listaDePrecos = new ArrayList();
        int contador = 1;

        try
        {
                pstmt = con.prepareStatement(SQL_CONSULTA_PRECOS);

                // Seta o MSISDN e o Codigo de Barras para a consulta
                pstmt.setString(1,msisdn);
                pstmt.setLong(2,Long.parseLong(codigo));

                // Executa a consulta
                rs = pstmt.executeQuery();

                // Enquanto o Resultado existir ou a quantidade for = 3
                while (rs.next() || contador <= Definitions.qtdeRegistros )
                {
                        listaDePrecos.add(getPreco(rs));
                        contador += 1;
                }
        }
        catch(SQLException e)
        {
                logger.error("Erro na consulta de precos. ERRO: ", e);
                throw e;
        }
        finally
        {
                if (rs != null)
                        rs.close();
        }
        return listaDePrecos;
}

public Product getPreco(ResultSet rs) throws SQLException
{
        // Novo objeto BarCode para conter os valores da consulta
        Product barcode = new Product();

        try
        {
                // Seta os valores da consulta no objeto
                barcode.setNoProduto(rs.getString("no_produto"));
                barcode.setNoLoja    (rs.getString("no_loja"));
                barcode.setVlPreco   (rs.getDouble("vl_preco"));
        }
        catch (SQLException e)
```

```
            {
                logger.error("Erro na selecao dos valores. ERRO: ", e);
                throw e;
            }
            // Retorno do resultado
            return barcode;
        }

        public void setERB (String loc)
        {
            // Somente os 4 ultimos bytes serao utilizados para a identificacao
            // do LAC e do CellID
            try
            {
                lac = Long.parseLong(getHexaString(loc.charAt(3))+getHexaString(loc.charAt(4)),16);
                cellId  = Long.parseLong(getHexaString(loc.charAt(5))+getHexaString(loc.charAt(6)),16);
            }
            catch(Exception e)
            {
                logger.error("Erro na concepcao da ERB. ERRO: ", e);
            }
        }

        private String getHexaString(char character)
        {
            String hexa = Integer.toHexString((int)character);
            try
            {
                hexa = df.format(Integer.parseInt(hexa));
            }
            catch (NumberFormatException n)
            {
                return "";
            }
            return hexa;
        }
}
```

- **Códigos-Fonte dos Objetos**

Abaixo são detalhados os códigos-fonte dos objetos utilizados no sistema. Têm como possuir apenas atributos e métodos *Set*, que serve para atribuir um valor para algum atributo e *Get*, que serve para resgatar o valor previamente atribuído a um determinado atributo.

```
// CLASSE AreaWalker
import br.com.codbarra.util.Definitions;
import br.com.codbarra.entity.RImage;
import java.util.Vector;

public class AreaWalker
{
   Point            startPoint;
   RImage           image;
   Vector           pendingPoints;
   public long added;
   public long removed;
```

```
    public AreaWalker(RImage im, Point p, boolean initVisited)
    {
        startPoint    = null;
        image         = null;
        pendingPoints = new Vector();
        added         = 0L;
        removed       = 0L;
        startPoint    = p;
        image         = im;
        if(initVisited)
            image.initializeVisited();

        pendingPoints.add(p);
    }

    public Point getNextPoint()
    {
        if(pendingPoints.size() == 0)
            return null;

        Point p = (Point)pendingPoints.elementAt(0);
        pendingPoints.removeElementAt(0);
        removed++;
        for(int i = 1; i <= 8; i++)
        {
            Point p2 = image.getNeightbour(p, i);

            if(p2 != null && !image.getVisited(p2))
            {
                image.setVisited(p2);
                if(image.getPixel(p2) == Definitions.FOREGROUND)
                {
                    added++;
                    pendingPoints.add(p2);
                }
            }
        }

        return p;
    }
}
```

// CLASSE Barcode1DObject

```
import java.util.Vector;
import br.com.codbarra.entity.Point;

public class Barcode1DObject
{
    public Vector bars;
    public Point corner1;
    public Point corner2;
    public Point corner3;
    public Point corner4;

    public Barcode1DObject(Vector bars)
    {
        this.bars = new Vector();
        this.bars = bars;
```

```
        corner1 = new Point(0.0D, 0.0D);
        corner2 = new Point(0.0D, 0.0D);
        corner3 = new Point(0.0D, 0.0D);
        corner4 = new Point(0.0D, 0.0D);

        ImageObject object1 = (ImageObject)bars.elementAt(0);
        ImageObject object2 = (ImageObject)bars.elementAt(bars.size() - 1);

        double angle = object1.getPA().getAngle() - 90D;
        double len = object1.getLength();
        double ystep = len * Math.cos(Math.toRadians(angle));
        double xstep = len * Math.sin(Math.toRadians(angle));

        corner1 = object1.nearestCorner;
        corner2 = new Point(Math.round(corner1.x - xstep), Math.round(corner1.y + ystep));
        corner3 = object2.farestCorner;
        corner4 = new Point(Math.round(corner3.x + xstep), Math.round(corner3.y - ystep));
    }

    public Vector getLines()
    {
        Vector v = new Vector();
        for(int i = 0; i < bars.size(); i++)
        {
            ImageObject object1 = (ImageObject)bars.elementAt(i);
            v.add(object1.getPA());
        }

        return v;
    }

    public double getOrientation()
    {
        return ((ImageObject)bars.elementAt(0)).getPA().getAngle();
    }

    public double getHeight()
    {
        return ((ImageObject)bars.elementAt(0)).getLength();
    }
}
```

// CLASSE BarcodeData

```
import br.com.codbarra.util.Definitions;

public class BarcodeData
{
    protected int symbology;
    public String value;
    public int x;
    public int y;

    public BarcodeData()
    {
        symbology = Definitions.CODE128;
        value = "";
        x = 0;
        y = 0;
    }
```

```
    public int getSymbology()
    {
        return symbology;
    }

    public void setSymbology(int s)
    {
        symbology = s;
    }

    public String getValue()
    {
        return value;
    }

    public void setValue(String v)
    {
        value = v;
    }

    public int getX()
    {
        return x;
    }

    public void setX(int i)
    {
        x = i;
    }

    public int getY()
    {
        return y;
    }

    public void setY(int i)
    {
        y = i;
    }

    public String toString()
    {
        String s = "Type: ";
        if(symbology == Definitions.CODE128)
            s = s + "CODE128";

        if(symbology == Definitions.EAN13)
            s = s + "EAN13";

        if(symbology == Definitions.IDENTCODE)
            s = s + "IDENTCODE";

        s = s + "\n";
        s = s + "Value: " + value + "\n";
        s = s + "X: " + x + "\n";
        s = s + "Y: " + y + "\n";
        return s;
    }
}
```

// CLASSE BarCodeImage

```
import java.awt.Image;

public class BarCodeImage
{
	private Image image;
	private String codeStr;
	private int code;
	private String description;

	public String getDescription()
	{
		return description;
	}

	public void setDescription(String description)
	{
		this.description = description;
	}

	public int getCode()
	{
		return code;
	}

	public void setCode(int code)
	{
		this.code = code;
	}

	public String getCodeStr()
	{
		return codeStr;
	}

	public void setCodeStr(String codeStr)
	{
		this.codeStr = codeStr;
	}

	public Image getImage()
	{
		return image;
	}

	public void setImage(Image image)
	{
		this.image = image;
	}
}
```

// CLASSE Client

```
public class Client
{
	private String		msisdn;
	private int		coCidade;
	private String noCidade;
```

```
	private Long  lac;
	private Long  cellId;

	public Long getCellId()
	{
		return cellId;
	}

	public void setCellId(Long cellId)
	{
		this.cellId = cellId;
	}

	public int getCoCidade()
	{
		return coCidade;
	}

	public void setCoCidade(int coCidade)
	{
		this.coCidade = coCidade;
	}

	public Long getLac()
	{
		return lac;
	}

	public void setLac(Long lac)
	{
		this.lac = lac;
	}

	public String getMsisdn()
	{
		return msisdn;
	}

	public void setMsisdn(String msisdn)
	{
		this.msisdn = msisdn;
	}

	public String getNoCidade()
	{
		return noCidade;
	}

	public void setNoCidade(String noCidade)
	{
		this.noCidade = noCidade;
	}
}
```

// CLASSE ImageObject

```
import br.com.codbarra.entity.AreaWalker;
import br.com.codbarra.entity.Line;
import br.com.codbarra.entity.Point;
import br.com.codbarra.entity.RImage;
```

```
import java.util.Vector;

public class ImageObject
{
   Point                startPoint;
   Point                centerOfMass;
   int          area;
   Line                 longPA;
   Line                 realPA;
   Point                perimeterPixels[];
   double               maxLength;
   Point                nearestCorner;
   double               nearestCornerDistance;
   double               farestCornerDistance;
   protected Point farestCorner;

   public ImageObject(RImage image, Point p)
   {
      startPoint = null;
      centerOfMass = null;
      area = 0;
      longPA = null;
      realPA = null;
      maxLength = 0.0D;
      nearestCorner = new Point(999999D, 999999D);
      nearestCornerDistance = 999999D;
      farestCorner = new Point(0.0D, 0.0D);
      farestCornerDistance = 0.0D;
      startPoint = p;
      initialize(image, p);
   }

   private void initialize(RImage image, Point start)
   {
      AreaWalker walker = new AreaWalker(image, start, false);
      double c1 = 0.0D;
      double c2 = 0.0D;
      area = 0;
      Vector v = new Vector();

      for(Point p = walker.getNextPoint(); p != null; p = walker.getNextPoint())
      {
         double distanceToOrigin = Math.sqrt(p.x * p.x + p.y * p.y);

         if(distanceToOrigin < nearestCornerDistance)
         {
            nearestCornerDistance = distanceToOrigin;
            nearestCorner = p;
         }

         if(distanceToOrigin > farestCornerDistance)
         {
            farestCornerDistance = distanceToOrigin;
            farestCorner = p;
         }
         area++;
         c1 += (int)p.y;
         c2 += (int)p.x;
         if(image.isInPerimeter4(p))
            v.add(p);
```

```
    }

    perimeterPixels = new Point[v.size()];
    for(int i = 0; i < v.size(); i++)
        perimeterPixels[i] = (Point)v.elementAt(i);

    centerOfMass = new Point(c2 / (double)area, c1 / (double)area);
}

public Point getStarPoint()
{
    return startPoint;
}

public Point getCenterOfMass()
{
    return centerOfMass;
}

public int getArea()
{
    return area;
}

public Line getPA()
{
    return realPA;
}

public void calculatePA(RImage image)
{
    if(longPA != null)
        return;

    double dmax = -1D;
    Point selected = null;

    for(int i = 0; i < perimeterPixels.length; i++)
    {
        Point p = perimeterPixels[i];

        if(p.y <= centerOfMass.y + 0.5D)
        {
            Line line = new Line(centerOfMass.x, centerOfMass.y, p.x, p.y);
            double dd = calculateDistances(image, line, dmax);

            if(dd < dmax || dmax == -1D)
            {
                dmax = dd;
                selected = p;
            }
        }
    }

    Line best = new Line(centerOfMass.x, centerOfMass.y, selected.x, selected.y);
    double len = selected.distanceTo(centerOfMass);
    dmax = -1D;
    Point selected2 = centerOfMass;
    double maxLen = 0.0D;
```

```
    for(int i = 0; i < perimeterPixels.length; i++)
    {
       Point p = perimeterPixels[i];

       if(p.y >= centerOfMass.y - 0.5D)
       {
          double pointLen = selected.distanceTo(p);

          if(pointLen > len)
          {
             Line line = new Line(selected.x, selected.y, p.x, p.y);
             double dd = calculateDistances(image, line, dmax);

             if(dd < dmax || dmax == -1D)
             {
                dmax = dd;
                selected2 = p;
             }
          }
       }
    }

    maxLen = 0.0D;
    realPA = new Line(selected.x, selected.y, selected2.x, selected2.y);
    best = realPA;

    for(int i = 0; i < perimeterPixels.length; i++)
    {
       Point p = perimeterPixels[i];

       if(p.y >= centerOfMass.y - 0.5D)
       {
          double pointLen = selected.distanceTo(p);

          if(pointLen > len)
          {
             double dist = best.getDistance(p);

             if(dist <= 1.0D && pointLen > maxLen)
             {
                selected2 = p;
                maxLen = pointLen;
             }
          }
       }
    }

    longPA = new Line(selected.x, selected.y, selected2.x, selected2.y);
    maxLength = longPA.getLength();
    perimeterPixels = null;
 }

 private double calculateMaxDistance(RImage image, Point p)
 {
    double max = 0.0D;

    for(int i = 0; i < perimeterPixels.length; i++)
    {
       Point p2 = perimeterPixels[i];
       double d = p.distanceTo(p2);
```

```
            if(d > max)
                max = d;
        }

        return max;
    }

    private double calculateDistances(RImage image, Line line, double currentMax)
    {
        double sum = 0.0D;

        for(int i = 0; i < perimeterPixels.length; i++)
        {
            Point p = perimeterPixels[i];
            sum += line.getDistance(p);

            if(currentMax != -1D && sum > currentMax)
                return currentMax + 1.0D;
        }

        return sum;
    }

    public boolean isAligned(ImageObject object)
    {
        Line perp = getPA().getPerp();
        double alignedDistance = perp.getDistance(object.getCenterOfMass());
        double tmp = 0.29999999999999999D * Math.abs(maxLength);

        return alignedDistance < tmp;
    }

    public double getLength()
    {
        return maxLength;
    }
}
```

// CLASSE Line

```
public class Line
{
    public double x1;
    public double y1;
    public double x2;
    public double y2;
    double                      slopeAngle;
    double                      slope;
    double                      length;
    protected double  a;
    protected double  b;
    protected double  c;
    public double       distanceToNextLine;
    public boolean      marked;

    public Line(Point p1, Point p2)
    {
        this(p1.x, p1.y, p2.x, p2.y);
    }
```

```
public Line(double x1, double y1, double x2, double y2)
{
   this.x1 = 0.0D;
   this.y1 = 0.0D;
   this.x2 = 0.0D;
   this.y2 = 0.0D;

   slopeAngle = 0.0D;
   slope = 0.0D;
   length = 0.0D;

   a = 0.0D;
   b = 0.0D;
   c = 0.0D;

   distanceToNextLine = 0.0D;
   marked = false;

   this.x1 = x1;
   this.x2 = x2;
   this.y1 = y1;
   this.y2 = y2;

   a = y2 - y1;
   b = x1 - x2;
   c = -(x1 - x2) * y1 + (y1 - y2) * x1;

   float distancex = (float)(x2 - x1);
   float distancey = (float)(y2 - y1);

   if(distancex == 0.0F)
   {
      slopeAngle = 90D;
      slope = 0.0D;
      length = Math.abs(distancey);
   }
   else
   {
     if(distancey == 0.0F)
           {
              slopeAngle = 0.0D;
              slope = 0.0D;
              length = Math.abs(distancex);
           }
     else
           {
              slopeAngle = Math.toDegrees(Math.atan(distancey / distancex));
              if(distancey < 0.0F && distancex < 0.0F)
                 slopeAngle = slopeAngle + 180D;

              if(distancey >= 0.0F && distancex < 0.0F)
                 slopeAngle = slopeAngle + 180D;

              slope = distancey / distancex;
              length = Math.sqrt(distancey * distancey + distancex * distancex);
           }
   }
}
```

```
    public Point getPointInSegment(double distance)
    {
        double middleX = x1 + distance * (x2 - x1);
        double middleY = y1 + distance * (y2 - y1);
        return new Point(middleX, middleY);
    }

    public Point getMiddlePoint()
    {
        double middleX = x1 + 0.5D * (x2 - x1);
        double middleY = y1 + 0.5D * (y2 - y1);
        return new Point(middleX, middleY);
    }

    public boolean isCenterAligned(Line line)
    {
        Line perp = getPerp();
        double alignedDistance = perp.getDistance(line.getMiddlePoint());
        double tmp = 0.10000000000000001D * Math.abs(line.getLength());
        return alignedDistance < tmp;
    }

    public Point getIntersect(Line line)
    {
        double dt = line.a * b - a * line.b;
        if(dt == 0.0D)
            return null;

        double y = Math.abs((line.a * c - a * line.c) / dt);
        double x = 0.0D;

        if(a == 0.0D)
            x = (-line.b / line.a) * y - line.c / line.a;
        else
            x = (-b / a) * y - c / a;

        return new Point(x, y);
    }

    public Line getPerp(double dist)
    {
        Point p = getMiddlePoint();
        double perpAngle = getAngle();
        perpAngle += 90D;
        if(perpAngle >= 360D)
            perpAngle -= 360D;

        return new Line(p, new Point((int)(p.x + dist * Math.cos(Math.toRadians(perpAngle))), (int)(p.y + dist *
Math.sin(Math.toRadians(perpAngle)))));
    }

    public Line getPerp(Point p, double dist)
    {
        double perpAngle = getAngle();
        perpAngle += 90D;

        if(perpAngle >= 360D)
            perpAngle -= 360D;
```

```
        return new Line(p, new Point((int)(p.x + dist * Math.cos(Math.toRadians(perpAngle))), (int)(p.y + dist *
Math.sin(Math.toRadians(perpAngle)))));
    }

    public double getLength()
    {
        return length;
    }

    public double getSlope()
    {
        return slope;
    }

    public double getAngle()
    {
        return slopeAngle;
    }

    public double getDistance(Point p)
    {
        return Math.abs(a * p.x + b * p.y + c) / Math.sqrt(a * a + b * b);
    }

    public Line getPerp()
    {
        return getPerp(500D);
    }

    public boolean isParallel(Line line, int h)
    {
        return false;
    }
}
```

// CLASSE Point

```
public class Point
{
    public double x;
    public double y;

    public Point(double x, double y)
    {
        this.x = 0.0D;
        this.y = 0.0D;
        this.x = x;
        this.y = y;
    }

    public double distanceTo(Point p)
    {
        return Math.sqrt((p.x - x) * (p.x - x) + (p.y - y) * (p.y - y));
    }

    public String toString()
    {
        return "(" + x + "," + y + ")";
    }
}
```

```
// CLASSE Product
public class Product
{
	private String noProduto;
	private String coProduto;
	private String noLoja;
	private double vlPreco;
	private int	coCidade;
	private String noCidade;

	public int getCoCidade()
	{
		return coCidade;
	}

	public void setCoCidade(int coCidade)
	{
		this.coCidade = coCidade;
	}

	public String getCoProduto()
	{
		return coProduto;
	}

	public void setCoProduto(String coProduto)
	{
		this.coProduto = coProduto;
	}

	public String getNoCidade()
	{
		return noCidade;
	}

	public void setNoCidade(String noCidade)
	{
		this.noCidade = noCidade;
	}

	public String getNoLoja()
	{
		return noLoja;
	}

	public void setNoLoja(String noLoja)
	{
		this.noLoja = noLoja;
	}

	public String getNoProduto()
	{
		return noProduto;
	}

	public void setNoProduto(String noProduto)
	{
		this.noProduto = noProduto;
```

```
        }

        public double getVlPreco()
        {
                return vlPreco;
        }

        public void setVlPreco(double vlPreco)
        {
                this.vlPreco = vlPreco;
        }
}
```

// CLASSE RImage

```
import br.com.codbarra.entity.Point;
import br.com.codbarra.util.Definitions;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.awt.image.WritableRaster;

public class RImage
{
   boolean visited[];
   int values[];
   BufferedImage image;
   int width;
   int height;

   public RImage(BufferedImage i)
   {
      visited = null;
      values  = null;
      image   = null;
      width     = 0;
      height    = 0;
      image     = i;
      setSize();
   }

   public RImage(BufferedImage i, Rectangle scanArea)
   {
      visited = null;
      values = null;
      image = null;
      width = 0;
      height = 0;
      BufferedImage input = new BufferedImage(scanArea.width, scanArea.height, i.getType());
      Graphics g = input.createGraphics();
      g.setColor(Color.white);
      g.fillRect(0, 0, scanArea.width, scanArea.height);
      g.drawImage(i, 0, 0, scanArea.width, scanArea.height, scanArea.x, scanArea.y, scanArea.x +
scanArea.width, scanArea.y + scanArea.height, null);
      g.dispose();
      g = null;
      image = input;
      setSize();
   }

   public RImage(int w, int h)
```

```
    {
        this(new BufferedImage(w, h, 12));
    }

    public void initializePixels()
    {
        if(values == null)
            imageToMemory();
    }

    private void imageToMemory()
    {
        WritableRaster raster = image.getRaster();
        int bands = raster.getNumBands();
        values = new int[height * width];

        if(bands == 1)
            raster.getSamples(0, 0, width, height, 0, values);
        else
        {
            raster.getSamples(0, 0, width, height, 3, values);
            int reds[] = raster.getSamples(0, 0, width, height, 1, (int[])null);
            int greens[] = raster.getSamples(0, 0, width, height, 2, (int[])null);

            for(int j = 0; j < height; j++)
            {
                for(int h = 0; h < width; h++)
                    values[h + j * width] = (reds[h + j * width] << 16) + (greens[h + j * width] << 8) + values[h + j *
width];
            }
        }
    }

    public void memoryToImage()
    {
        WritableRaster raster = image.getRaster();
        int bands = raster.getNumBands();
        if(bands == 1)
            raster.setSamples(0, 0, width, height, 0, values);
        else
        {
            for(int j = 0; j < height; j++)
            {
                for(int h = 0; h < width; h++)
                {
                    raster.setSample(h, j, 1, getRPixel(j, h));
                    raster.setSample(h, j, 2, getGPixel(j, h));
                    raster.setSample(h, j, 3, getBPixel(j, h));
                }
            }
        }
    }

    private BufferedImage getImageCopy(BufferedImage image)
    {
        BufferedImage input = new BufferedImage(image.getWidth(null), image.getHeight(null), 2);
        Graphics g = input.createGraphics();
        g.setColor(Color.white);
        g.fillRect(0, 0, image.getWidth(), image.getHeight());
        g.drawImage(image, 0, 0, null);
```

```
        g.dispose();
        g = null;
        return input;
    }

    public void resetImage()
    {
        imageToMemory();
    }

    private void setSize()
    {
        width = image.getWidth();
        height = image.getHeight();
    }

    public int getWidth()
    {
        return width;
    }

    public int getHeight()
    {
        return height;
    }

    public BufferedImage getImage()
    {
        return image;
    }

    public void setPixel(int row, int col, int val)
    {
        try
        {
            values[col + row * width] = val;
        }
        catch(Exception e)
        {
            e.printStackTrace();
            System.err.println("row = " + row + " col=" + col);
            System.err.println("h = " + height + " w=" + width);
            throw new RuntimeException(e);
        }
    }

    public void setPixel(Point p, int val)
    {
        setPixel((int)p.y, (int)p.x, val);
    }

    public int getGreyPixel(double row, double col)
    {
        int y = (int)Math.abs(row);
        int x = (int)Math.abs(col);
        return (int)((double)getRPixel(y, x) * 0.33000000000000002D + (double)getGPixel(y, x) *
0.33000000000000002D + (double)getBPixel(y, x) * 0.33000000000000002D + 0.5D);
    }

    public int getPixel(Point p)
```

```
{
   if(p == null)
      return Definitions.BACKGROUND;
   else
      return getPixel(p.y, p.x);
}

public int getPixel(double row, double col)
{
   return getPixel((int)Math.round(row), (int)Math.round(col));
}

public int getPixel(int row, int col)
{
   return values[col + row * width];
}

public int getRPixel(int row, int col)
{
   return (getPixel(row, col) & 0xff0000) >> 16;
}

public void setRPixel(int row, int col, int val)
{
   int cur = getPixel(row, col);
   cur = cur & 0xffff | val << 16;
   setPixel(row, col, cur);
}

public int getGPixel(int row, int col)
{
   return (getPixel(row, col) & 0xff00) >> 8;
}

public void setGPixel(int row, int col, int val)
{
   int cur = getPixel(row, col);
   cur = cur & 0xff00ff | val << 8;
   setPixel(row, col, cur);
}

public int getBPixel(int row, int col)
{
   return getPixel(row, col) & 0xff;
}

public void setBPixel(int row, int col, int val)
{
   int cur = getPixel(row, col);
   cur = cur & 0xffff00 | val;
   setPixel(row, col, cur);
}

public void initializeVisited()
{
   visited = new boolean[width * height];
}

public void setVisited(Point p)
{
```

```
    setVisited((int)p.x, (int)p.y);
}

public void setVisited(int x, int y)
{
    visited[x + y * width] = true;
}

public boolean getVisited(Point p)
{
    return getVisited((int)p.x, (int)p.y);
}

public boolean getVisited(int x, int y)
{
    return visited[x + y * width];
}

public boolean isInPerimeter4(Point point)
{
    if(getPixel(getNeightbour(point, 1)) == Definitions.BACKGROUND)
        return true;

    if(getPixel(getNeightbour(point, 3)) == Definitions.BACKGROUND)
        return true;

    if(getPixel(getNeightbour(point, 5)) == Definitions.BACKGROUND)
        return true;

    return getPixel(getNeightbour(point, 7)) == Definitions.BACKGROUND;
}

public Point getNeightbour(Point point, int p)
{
    if(p == 1 && point.x < (double)(getWidth() - 1))
        return new Point(point.x + 1.0D, point.y);

    if(p == 8 && point.x < (double)(getWidth() - 1) && point.y > 0.0D)
        return new Point(point.x + 1.0D, point.y - 1.0D);

    if(p == 7 && point.y > 0.0D)
        return new Point(point.x, point.y - 1.0D);

    if(p == 6 && point.x > 0.0D && point.y > 0.0D)
        return new Point(point.x - 1.0D, point.y - 1.0D);

    if(p == 5 && point.x > 0.0D)
        return new Point(point.x - 1.0D, point.y);

    if(p == 4 && point.x > 0.0D && point.y < (double)(getHeight() - 1))
        return new Point(point.x - 1.0D, point.y + 1.0D);

    if(p == 3 && point.y < (double)(getHeight() - 1))
        return new Point(point.x, point.y + 1.0D);

    if(p == 2 && point.y < (double)(getHeight() - 1) && point.x < (double)(getWidth() - 1))
        return new Point(point.x + 1.0D, point.y + 1.0D);
    else
        return null;
}
```

```
}


// CLASSE VectorizedImage
import java.awt.Color;
import java.awt.Graphics;
import java.util.Vector;
import br.com.codbarra.entity.Line;

public class VectorizedImage
{
   protected Vector lines;
   protected Vector objects;

   private void render(Graphics g, Vector pLines, int x, int y)
   {
      g.setColor(Color.black);
      Color colors[] = {
         Color.red, Color.green, Color.blue, Color.yellow, Color.cyan, Color.orange, Color.pink
      };
      for(int i = 0; i < pLines.size(); i++)
      {
         Line l = (Line)pLines.elementAt(i);
         g.setColor(colors[i % colors.length]);
         g.drawLine((int)l.x1 - x, (int)l.y1 - y, (int)l.x2 - x, (int)l.y2 - y);
      }
   }

   public void render(Graphics g)
   {
      render(g, lines, 0, 0);
   }

   public VectorizedImage()
   {
      lines = new Vector();
      objects = new Vector();
   }

   public void addLine(Line l)
   {
      lines.add(l);
   }

   public void addObject(ImageObject l)
   {
      objects.add(l);
      lines.add(l.getPA());
   }

   public Vector getLines()
   {
      return lines;
   }

   public Vector getObjects()
   {
      return objects;
   }
}
```

- **Códigos-Fonte dos Utilitários**

A seguir são detalhados os códigos-fonte dos utilitários, ou seja, classes que fazer a devida interpretação dos códigos de barras para a extração do código numérico. Há também o tratamento das imagens, através da transformação para o padrão preto e branco, o que facilita a interpretação das mesmas.

```
// CLASSE BarcodeParser
public class BarcodeParser
{
   protected double  barcode[];
   protected int pointer;
   protected String code;
   protected String startPatterns[] = { "11" };
   protected String endPattern;
   protected int maxModuleSize;
   protected int modulesProCharacter;
   protected double knownBarPatterns[];
   protected double knownSpacePatterns[];
   protected boolean initializedPatterns;
   public boolean checkSumExpected;

   public BarcodeParser()
   {
      barcode = null;
      pointer = 0;
      code = "";
      endPattern = "11";
      maxModuleSize = 4;
      modulesProCharacter = 9;
      knownBarPatterns = new double[10];
      knownSpacePatterns = new double[10];
      initializedPatterns = false;
      checkSumExpected = true;
   }

   public String getParsedCode()
   {
      return code;
   }

   public void init(double widths[])
   {
      barcode = widths;
   }

   public boolean parse()
   {
      return true;
   }

   protected void reverseBarcode()
   {
      double reversed[] = new double[barcode.length];
```

```
   for(int i = 0; i < barcode.length; i++)
      reversed[barcode.length - 1 - i] = barcode[i];

   resetPointer();
   barcode = reversed;
}

protected int getPatternIndex(String set[], String pattern)
{
   for(int i = 0; i < set.length; i++)
   {
      if(set[i].equals(pattern))
         return i;
   }

   return -1;
}

protected int getPatternIndex(String set[][], int index, String pattern)
{
   for(int i = 0; i < set.length; i++)
   {
      if(set[i][index].equals(pattern))
         return i;
   }

   return -1;
}

protected String getPattern(int numberOfBars, boolean movePointer, String allowedSet[])
{
   int j = pointer;
   String pattern = "";
   double errors[] = new double[numberOfBars];
   int moduleSum = 0;

   if(!initializePatterns())
      return "";

   for(int i = 0; i < numberOfBars && j < barcode.length; i++)
   {
      if(movePointer)
         pointer++;

      double val = barcode[j];
      double knownPatterns[] = knownBarPatterns;
      if(j % 2 == 1)
         knownPatterns = knownSpacePatterns;

      if(val <= knownPatterns[1])
      {
         pattern = pattern + "1";
         moduleSum++;
      }
      else
            if(val > knownPatterns[maxModuleSize])
             {
                pattern = pattern + maxModuleSize;
                errors[i] = val - knownPatterns[maxModuleSize];
                moduleSum += maxModuleSize;
```

```
            } else
            {
               for(int h = 1; h < maxModuleSize; h++)
               {
                  double dist = (knownPatterns[h + 1] - knownPatterns[h]) / 2D;
                  if(val <= knownPatterns[h + 1] && val >= knownPatterns[h + 1] - dist)
                  {
                     pattern = pattern + (h + 1);
                     errors[i] = val - knownPatterns[h + 1];
                     moduleSum = moduleSum + h + 1;
                     break;
                  }
                  if(val < knownPatterns[h] || val > knownPatterns[h] + dist)
                     continue;

                  pattern = pattern + h;
                  errors[i] = val - knownPatterns[h];
                  moduleSum += h;
                  break;
               }
            }

      j++;
   }

   if(allowedSet.length > 1)
   {
      while(moduleSum != modulesProCharacter)
      {
         int diff = modulesProCharacter - moduleSum;
         int candidate = -1;

         for(int i = 0; i < pattern.length(); i++)
         {
            if(errors[i] != 0.0D)
            {
               int currentModule = (new Integer("" + pattern.charAt(i))).intValue();
               if(diff < 0 && currentModule > 1)
               {
                  if(candidate == -1)
                     candidate = i;

                  if(errors[i] < errors[candidate])
                     candidate = i;
               }
               if(diff > 0 && currentModule < maxModuleSize)
               {
                  if(candidate == -1)
                     candidate = i;

                  if(errors[i] > errors[candidate])
                     candidate = i;
               }
            }
         }

         if(candidate < 0)
            break;

         int module = (new Integer("" + pattern.charAt(candidate))).intValue();
```

```
            errors[candidate] = 0.0D;
            if(diff > 0)
            {
               module++;
               moduleSum++;
            }
            else
            {
               module--;
               moduleSum--;
            }

            pattern = pattern.substring(0, candidate) + module + pattern.substring(candidate + 1);
         }
      }
      return pattern;
   }

   protected boolean initializePatterns()
   {
      if(initializedPatterns)
         return true;

      initializedPatterns = true;
      String knownPattern = "";
      double receivedPattern[] = null;
      if(startPatterns.length == 1)
      {
         knownPattern = startPatterns[0];
         receivedPattern = new double[knownPattern.length()];
         for(int i = 0; i < knownPattern.length(); i++)
            receivedPattern[i] = barcode[i];
      }
      else
      {
         knownPattern = endPattern;
         receivedPattern = new double[knownPattern.length()];
         for(int i = 0; i < knownPattern.length(); i++)
            receivedPattern[i] = barcode[(barcode.length - knownPattern.length()) + i];
      }

      for(int i = 0; i < knownPattern.length(); i++)
      {
         int currentModule = (new Integer("" + knownPattern.charAt(i))).intValue();
         double knownPatterns[] = knownBarPatterns;
         if(i % 2 == 1)
            knownPatterns = knownSpacePatterns;

         if(knownPatterns[currentModule] == 0.0D)
            knownPatterns[currentModule] = receivedPattern[i];
         else
         {
            double diff = knownPatterns[currentModule] - receivedPattern[i];

            if(Math.abs(diff / knownPatterns[currentModule]) > 0.5D)
               return false;
         }

         if(currentModule > 1 && knownPatterns[0] > 0.0D)
         {
```

```
            double diff = knownPatterns[currentModule] - receivedPattern[i] / knownPatterns[0];
            if(Math.abs(diff / knownPatterns[currentModule]) > 0.5D)
               return false;
         }
      }

      completePattern();
      return true;
   }

   private void completePattern()
   {
      double correction = knownSpacePatterns[1] - knownBarPatterns[1];
      if(Math.abs(correction) > 0.75D)
         correction /= 2D;
      else
         correction = 0.0D;

      double p[] = knownBarPatterns;

      if(p[2] == 0.0D && p[3] > 0.0D)
         p[2] = p[1] + (p[3] - p[1]) / 2D;

      if(p[2] == 0.0D)
         p[2] = (p[1] + correction) * 2D - correction;

      if(p[3] == 0.0D)
         p[3] = (p[1] + correction) * 3D - correction;

      if(p[4] == 0.0D)
         p[4] = p[3] + (p[2] - p[1]);

      p = knownSpacePatterns;
      if(p[2] == 0.0D && p[3] > 0.0D)
         p[2] = p[1] + (p[3] - p[1]) / 2D;

      if(p[2] == 0.0D)
         p[2] = (p[1] - correction) * 2D + correction;

      if(p[3] == 0.0D)
         p[3] = (p[1] - correction) * 3D + correction;

      if(p[4] == 0.0D)
         p[4] = p[3] + (p[2] - p[1]);
   }

   protected void resetPointer()
   {
      pointer = 0;
      initializedPatterns = false;
      knownBarPatterns = new double[10];
      knownSpacePatterns = new double[10];
   }

   protected boolean endReached()
   {
      return pointer >= barcode.length;
   }

   protected void addToCode(String s)
```

```
    {
        if(s.equals("~"))
            code = code + "~";

        code = code + s;
    }

    protected void addToCode(int s)
    {
        String tmp = "" + s;
        if(tmp.length() < 3)
            tmp = "0" + tmp;

        if(tmp.length() < 3)
            tmp = "0" + tmp;

        code = code + "~d" + tmp;
    }

    public int getSymbology()
    {
        return 0;
    }

    private double getLeastSquares(int values[])
    {
        int pts = values.length;
        int degree = 1;
        RMatrix Pm = new RMatrix(pts, degree + 1);
        RMatrix Ym = new RMatrix(pts, 1);
        double minx = values[0];
        double maxx = values[0];

        for(int i = 0; i < pts; i++)
        {
            if((double)values[i] < minx)
                minx = values[i];

            if((double)values[i] > maxx)
                maxx = values[i];

            for(int k = 0; k <= degree; k++)
                Pm.setValue(i, k, Math.pow(values[i], k));

            Ym.setValue(i, 0, values[i]);
        }

        RMatrix T = Pm.transpose();
        RMatrix MT = T.mult(Pm);
        RMatrix resultMatrix = MT.invert().mult(T).mult(Ym);
        double a1 = resultMatrix.getValue(0, 0);
        double a2 = resultMatrix.getValue(1, 0);
        int x1 = values[0];
        int y1 = (int)Math.round(a1 + (double)x1 * a2);
        //int x2 = values[pts - 1];
        //int y2 = (int)Math.round(a1 + (double)x2 * a2);
        return (double)y1;
    }
}
```

```
// CLASSE CodeEAN13Parser
import org.apache.log4j.Logger;
import br.com.codbarra.util.Definitions;
import br.com.codbarra.util.Validator;

public class CodeEAN13Parser extends BarcodeParser
{
        protected static String START  = "111"  ;
   protected static String STOP   = "111"  ;
   protected static String CENTER = "11111";
   private Logger logger = Logger.getLogger(this.getClass());

   protected String setEANLeftA[][] =
   {
                {"0", "3211"},{"1", "2221"},{"2", "2122"},{"3", "1411"},{"4", "1132"},
                {"5", "1231"},{"6", "1114"},{"7", "1312"},{"8", "1213"},{"9", "3112"}
   };

   protected String setEANLeftB[][] =
   {
                {"0", "1123"},{"1", "1222"},{"2", "2212"},{"3", "1141"},{"4", "2311"},
                {"5", "1321"},{"6", "4111"},{"7", "2131"},{"8", "3121"}, {"9", "2113"}
   };

   protected String setEANRight[][] =
   {
                {"0", "3211"},{"1", "2221"},{"2", "2122"},{"3", "1411"},{"4", "1132"},
                {"5", "1231"},{"6", "1114"},{"7", "1312"},{"8", "1213"},{"9", "3112"}
   };

   protected String setEANCode[] = {"AAAAA", "ABABB", "ABBAB", "ABBBA", "BAABB", "BBAAB",
"BBBAA", "BABAB", "BABBA", "BBABA"};

   String eanPatternLeft[];
   String eanPatternRight[];

   public CodeEAN13Parser()
   {
      eanPatternLeft = null;
      eanPatternRight = null;
      super.startPatterns = (new String[] { START } );

      super.endPattern = STOP;
      super.maxModuleSize = 4;
      eanPatternLeft = new String[setEANLeftA.length + setEANLeftB.length];
      eanPatternRight = new String[setEANRight.length];
      int counter = 0;

      for(int i = 0; i < setEANLeftA.length; i++)
         eanPatternLeft[counter++] = setEANLeftA[i][1];

      for(int i = 0; i < setEANLeftB.length; i++)
         eanPatternLeft[counter++] = setEANLeftB[i][1];

      counter = 0;

      for(int i = 0; i < setEANRight.length; i++)
         eanPatternRight[counter++] = setEANRight[i][1];
```

```
    super.modulesProCharacter = 7;
}

public boolean parse()
{
    super.code = "";
    super.resetPointer();
    boolean r = parse13();
    if(!r)
    {
        reverseBarcode();
        r = parse13();
    }

    return r;
}

public boolean parse13()
{
    String parity = "";
    String pattern = getPattern(3, true, new String[] { START } );

    if(!pattern.equals(START))
    {
        logger.debug("Inicio do padrao de codigo EAN 13 nao reconhecido: " + pattern);
        return false;
    }

    for(int i = 0; i < 6; i++)
    {
        pattern = getPattern(4, true, eanPatternLeft);
        int index = -1;
        index = super.getPatternIndex(setEANLeftA, 1, pattern);
        if(index == -1)
        {
            if(i > 0)
                index = super.getPatternIndex(setEANLeftB, 1, pattern);
            if(index == -1)
            {
                logger.debug("Padrao nao reconhecido: " + pattern);
                return false;
            }
            if(i > 0)
                parity = parity + "B";
        }
        else
        if(i > 0)
            parity = parity + "A";

        addToCode(setEANLeftA[index][0]);
    }

    int parityIndex = super.getPatternIndex(setEANCode, parity);
    super.code = "" + parityIndex + super.code;
    pattern = getPattern(5, true, new String[] { CENTER } );

    if(!pattern.equals(CENTER))
    {
        logger.debug("Centro de gravidade padrao do codigo EAN 13 nao reconhecido: " + pattern);
        return false;
```

```
        }

        for(int i = 0; i < 6; i++)
        {
           pattern = getPattern(4, true, eanPatternRight);
           int index = -1;
           index = super.getPatternIndex(setEANRight, 1, pattern);
           if(index == -1)
           {
                 logger.debug("Padrao nao reconhecido: " + pattern);
              return false;
           }
           String c = setEANRight[index][0];
           if(i == 5)
           {
                 String checksum = Validator.getDigVerificador(super.code);
              //String checksum = UPCEANCheck(super.code);
              if(!c.equals(checksum))
                 logger.debug("O calculo da soma esta incorreto: " + c + " <> " + checksum);
           }
           addToCode(c);
        }

        pattern = getPattern(3, true, new String[] { STOP } );

        if(!pattern.equals(STOP))
        {
          logger.debug("Final do padrao do codigo EAN 13 nao reconhecido: " + pattern);
           return false;
        }
        else
           return true;
     }

     public int getSymbology()
     {
        return Definitions.EAN13;
     }
}


// CLASSE RMatrix
class RMatrix
{
   private int rows;
   private int columns;
   private double data[][];

   public RMatrix(double d[][])
   {
      rows = d.length;
      columns = d[0].length;
      data = d;
   }

   public RMatrix(int d[][])
   {
      rows = d.length;
      columns = d[0].length;
      data = new double[rows][columns];
```

```
    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < columns; j++)
            data[i][j] = d[i][j];
    }
}

public RMatrix(int prows, int pcolumns)
{
    rows = prows;
    columns = pcolumns;
    data = new double[rows][columns];
    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < columns; j++)
            data[i][j] = 0.0D;
    }
}

public RMatrix add(RMatrix m2)
{
    RMatrix newMatrix = new RMatrix(rows, columns);
    if(rows == m2.rows && columns == m2.columns)
    {
        for(int i = 0; i < rows; i++)
        {
            for(int j = 0; j < columns; j++)
                newMatrix.data[i][j] = data[i][j] + m2.data[i][j];
        }
    }

    return newMatrix;
}

public RMatrix invert()
{
    int n = data.length;
    double D[][] = new double[n + 1][2 * n + 2];
    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < columns; j++)
            D[i + 1][j + 1] = data[i][j];
    }

    //int error = 0;
    int n2 = 2 * n;
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
            D[i][j + n] = 0.0D;

        D[i][i + n] = 1.0D;
    }

    for(int i = 1; i <= n; i++)
    {
        double alpha = D[i][i];
        if(alpha == 0.0D)
        {
            //error = 1;
```

```
            break;
        }
        for(int j = 1; j <= n2; j++)
            D[i][j] = D[i][j] / alpha;

        for(int k = 1; k <= n; k++)
        {
            if(k - i != 0)
            {
                double beta = D[k][i];
                for(int j = 1; j <= n2; j++)
                    D[k][j] = D[k][j] - beta * D[i][j];
            }
        }
    }

    RMatrix m = new RMatrix(rows, columns);
    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < columns; j++)
            m.data[i][j] = D[i + 1][j + 1 + n];
    }

    return m;
}

public RMatrix mult(double v)
{
    RMatrix newMatrix = new RMatrix(rows, columns);

    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < columns; j++)
            newMatrix.data[i][j] = v * data[i][j];
    }

    return newMatrix;
}

public void setValue(int r, int c, double v)
{
    data[r][c] = v;
}

public double getValue(int r, int c)
{
    return data[r][c];
}

public RMatrix mult(RMatrix m2)
{
    RMatrix m1 = this;
    RMatrix newMatrix = new RMatrix(m1.rows, m2.columns);
    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < m2.columns; j++)
        {
            for(int k = 0; k < columns; k++)
                newMatrix.data[i][j] += data[i][k] * m2.data[k][j];
        }
```

```
        }

        return newMatrix;
    }

    public RMatrix transpose()
    {
        RMatrix newMatrix = new RMatrix(columns, rows);
        for(int i = 0; i < rows; i++)
        {
            for(int j = 0; j < columns; j++)
                newMatrix.data[j][i] = data[i][j];
        }

        return newMatrix;
    }
}
```

// CLASSE Barcode1DFinder

```
import java.util.Vector;
import org.apache.log4j.Logger;
import br.com.codbarra.entity.Barcode1DObject;
import br.com.codbarra.entity.Line;
import br.com.codbarra.entity.ImageObject;
import br.com.codbarra.entity.Point;
import br.com.codbarra.entity.VectorizedImage;

public class Barcode1DFinder
{
    public int          minbars;
    public int          minBarLength;
    public int          quiteZoneMultiplier;
    public boolean supportBrokenBars;
    private Logger logger = Logger.getLogger(this.getClass());

    public Barcode1DFinder()
    {
        minbars = 10;
        minBarLength = 15;
        quiteZoneMultiplier = 10;
        supportBrokenBars = true;
    }

    public Vector findBarcodes(VectorizedImage image)
    {
        Vector objects = image.getObjects();
        Vector candidates = new Vector();

        while(objects.size() > 0)
        {
            Vector unsorted = getParallelObjectsSet(objects);
            Vector set = sortParallelAlignedObjects(unsorted);
            calculateDistances(set);

            if(set.size() > minbars)
            {
                Vector groups = getGroupsOfObjects(set);
                for(int i = 0; i < groups.size(); i++)
                    candidates.add(new Barcode1DObject((Vector)groups.elementAt(i)));
```

```
            }
        }

        return candidates;
    }

    private Vector sortParallelAlignedObjects(Vector objects)
    {
        Line first = ((ImageObject)objects.elementAt(0)).getPA();
        Line xAxis = new Line(0.0D, 0.0D, 1.0D, 0.0D);
        Line yAxis = new Line(0.0D, 0.0D, 0.0D, 1.0D);
        Line perp = first.getPerp();
        Line axis = xAxis;
        if(Math.abs(perp.getAngle() % 360D) < 25D)
            axis = yAxis;

        if(Math.abs((perp.getAngle() + 180D) % 360D) < 25D)
            axis = yAxis;

        Vector distances = new Vector();
        Vector sorted = new Vector();
        Point referencePoint = axis.getIntersect(perp);

        for(int i = 0; i < objects.size(); i++)
        {
            ImageObject object = (ImageObject)objects.elementAt(i);
            Line line = object.getPA();
            double dist = line.getDistance(referencePoint);
            int position = distances.size();
            for(int j = 0; j < distances.size(); j++)
            {
                if(((Double)distances.elementAt(j)).doubleValue() <= dist)
                    continue;

                position = j;
                break;
            }

            distances.add(position, new Double(dist));
            sorted.add(position, object);
        }

        int j = distances.size() - 1;
        double firstBarWidth = (double)((ImageObject)sorted.elementAt(0)).getArea() /
((ImageObject)sorted.elementAt(0)).getLength();
        double minDistance = firstBarWidth * 0.5D;

        if(minDistance < 1.5D)
            minDistance = 1.5D;

        if(supportBrokenBars)
        {
            for(; j > 0; j--)
            {
                double previous = ((Double)distances.elementAt(j - 1)).doubleValue();
                double current = ((Double)distances.elementAt(j)).doubleValue();
                if(current - previous < minDistance)
                {
                    ImageObject pobject = (ImageObject)sorted.elementAt(j - 1);
                    ImageObject cobject = (ImageObject)sorted.elementAt(j);
```

```
                if(cobject.getLength() > pobject.getLength())
                {
                    distances.removeElementAt(j - 1);
                    sorted.removeElementAt(j - 1);
                }
                else
                {
                    distances.removeElementAt(j);
                    sorted.removeElementAt(j);
                }
            }
        }
    }

    return sorted;
}

private boolean compareAngles(double a1, double a2, double diff)
{
    if(Math.abs(a1 - a2) < diff)
        return true;

    double tmp = a1 + 180D;

    if(tmp > 360D)
        tmp -= 360D;

    if(Math.abs(tmp - a2) < diff)
        return true;

    tmp = a2 + 180D;

    if(tmp > 360D)
        tmp -= 360D;

    return Math.abs(a1 - tmp) < diff;
}

private Vector getParallelObjectsSet(Vector objects)
{
    Vector result = new Vector();
    Vector resultIndexes = new Vector();
    ImageObject first = (ImageObject)objects.elementAt(0);
    result.add(first);
    resultIndexes.add(new Integer(0));

    for(int i = 1; i < objects.size(); i++)
    {
        ImageObject object = (ImageObject)objects.elementAt(i);
        Line line = object.getPA();
        if(compareAngles(line.getAngle(), first.getPA().getAngle(), 3D))
        {
            if(object.getLength() > (double)minBarLength)
            {
                if(first.isAligned(object))
                {
                    result.add(object);
                    resultIndexes.add(new Integer(i));
                }
                else
```

```
                    logger.debug("Nao alinhado.");
                }
                else
                    logger.debug("Nao longo suficiente " + object.getLength());
            }
            else
                    logger.debug("Angulo diferente " + object.getPA().getAngle() + "  <> " + first.getPA().getAngle()
+ " tamanho " + first.getLength());
        }

        for(int i = resultIndexes.size() - 1; i >= 0; i--)
            objects.removeElementAt(((Integer)resultIndexes.elementAt(i)).intValue());

        return result;
    }

    private void calculateDistances(Vector result)
    {
        for(int i = 0; i < result.size() - 1; i++)
        {
            Line line = ((ImageObject)result.elementAt(i)).getPA();
            Line line2 = ((ImageObject)result.elementAt(i + 1)).getPA();
            line2.distanceToNextLine = 0.0D;
            line.distanceToNextLine = line.getDistance(new Point(line2.x1, line2.y1));
        }
    }

    private Vector getGroupsOfObjects(Vector parallelObjects)
    {
        int minLine = getMinDistanceLine(parallelObjects);
        Vector groups = new Vector();

        for(; minLine != -1; minLine = getMinDistanceLine(parallelObjects))
        {
            Vector group = new Vector();
            double minDistance = ((ImageObject)parallelObjects.elementAt(minLine)).getPA().distanceToNextLine;
            //int startLine = minLine;
            for(int i = minLine - 1; i >= 0; i--)
            {
                ImageObject object = (ImageObject)parallelObjects.elementAt(i);
                Line line = ((ImageObject)parallelObjects.elementAt(i)).getPA();

                if(line.marked || line.distanceToNextLine > minDistance * (double)quiteZoneMultiplier)
                    break;

                line.marked = true;
                group.add(0, object);
                //startLine = i;
            }

            //int endLine = minLine;
            for(int i = minLine; i < parallelObjects.size(); i++)
            {
                ImageObject object = (ImageObject)parallelObjects.elementAt(i);
                Line line = ((ImageObject)parallelObjects.elementAt(i)).getPA();

                if(line.marked)
                    break;

                line.marked = true;
```

```
            group.add(object);
            //endLine = i;

            if(line.distanceToNextLine > minDistance * (double)quiteZoneMultiplier)
               break;
         }

         if(group.size() >= minbars)
            groups.add(group);
      }

      return groups;
   }

   private int getMinDistanceLine(Vector objects)
   {
      double min = 999999D;
      int minLine = -1;

      for(int i = 0; i < objects.size(); i++)
      {
         Line line = ((ImageObject)objects.elementAt(i)).getPA();
         if(!line.marked && min > line.distanceToNextLine && line.distanceToNextLine > 0.0D)
         {
            minLine = i;
            min = line.distanceToNextLine;
         }
      }

      return minLine;
   }
}
```

// CLASSE BarsReader

```
import br.com.codbarra.entity.Barcode1DObject;
import br.com.codbarra.entity.ImageObject;
import br.com.codbarra.entity.Line;
import br.com.codbarra.entity.Point;
import br.com.codbarra.entity.RImage;
import br.com.codbarra.util.Definitions;
import java.awt.Polygon;
import java.util.Vector;
import org.apache.log4j.Logger;

public class BarsReader
{
   public static int ALG_AREAS = 0;
   public static int ALG_LINE = 1;
   public static int ALG_HISTOGRAM = 2;
   private Logger logger = Logger.getLogger(this.getClass());

   public BarsReader()
   {
   }

   public double[] convertBarsToWidths(int algorithm, RImage image, Barcode1DObject barcode)
   {
      if(algorithm == ALG_AREAS)
         return convertAreasToBars(image, barcode);
```

```
    if(algorithm == ALG_HISTOGRAM)
      return convertHistogramToBars(image, barcode);
    else
      return convertCentralLineToBars(image, barcode);
  }

  private double[] convertHistogramToBars(RImage image, Barcode1DObject barcode)
  {
    Vector values = new Vector();
    double counter = 0.0D;
    double totalCounter = 0.0D;

    Point startPoint = barcode.corner2;
    Point endPoint = barcode.corner3;

    double angle = barcode.getOrientation() + 90D;
    double len = barcode.getHeight();
    double ystep = len * Math.cos(Math.toRadians(angle));
    double xstep = len * Math.sin(Math.toRadians(angle));

    LineWalker walker = new LineWalker((int)startPoint.x, (int)startPoint.y, (int)endPoint.x, (int)endPoint.y);
    Point p = walker.getNextPoint();
    Point previousPointH = p;

    for(; p != null; p = walker.getNextPoint())
    {
      if(Math.round(p.x) != Math.round(previousPointH.x) || Math.round(p.y) !=
Math.round(previousPointH.y))
      {
        Point basePoint = p;
        Point verticalEnd = new Point(Math.round(basePoint.x - xstep), Math.round(basePoint.y + ystep));
        LineWalker verticalWalker = new LineWalker((int)basePoint.x, (int)basePoint.y, (int)verticalEnd.x,
(int)verticalEnd.y);
        counter = 0.0D;
        totalCounter = 0.0D;
        Point p2 = verticalWalker.getNextPoint();
        Point previousPointV = p2;
        for(; p2 != null; p2 = verticalWalker.getNextPoint())
        {
          if(Math.round(p2.x) != Math.round(previousPointV.x) || Math.round(p2.y) !=
Math.round(previousPointV.y))
          {
            totalCounter++;
            int value = image.getPixel(p2);

            if(value == Definitions.FOREGROUND)
              counter++;
          }
          previousPointV = p2;
        }

        values.add(new Double(counter / totalCounter));
      }
      previousPointH = p;
    }

    logger.debug("");
    for(int i = 0; i < values.size(); i++)
      logger.debug(((Double)values.elementAt(i)).doubleValue() + " , ");
```

```
    logger.debug("");

    return null;
}

private double[] convertCentralLineToBars(RImage image, Barcode1DObject barcode)
{
    Vector result = new Vector();
    Vector values = new Vector();
    int counter = 0;
    int currentValue = 0;
    Point startPoint = (new Line(barcode.corner1, barcode.corner2)).getMiddlePoint();
    Point endPoint = (new Line(barcode.corner3, barcode.corner4)).getMiddlePoint();
    LineWalker walker = new LineWalker((int)startPoint.x, (int)startPoint.y, (int)endPoint.x, (int)endPoint.y);
    Point p = walker.getNextPoint();
    Point previousPoint = p;
    currentValue = image.getPixel(p);
    counter++;

    for(; p != null; p = walker.getNextPoint())
    {
        if(Math.round(p.x) != Math.round(previousPoint.x) || Math.round(p.y) != Math.round(previousPoint.y))
        {
            int value = image.getPixel(p);
            if(value != currentValue)
            {
                values.add(new Integer(currentValue));
                result.add(new Integer(counter));
                currentValue = value;
                counter = 1;
            } else
                counter++;
        }

        previousPoint = p;
    }

    values.add(new Integer(currentValue));
    result.add(new Integer(counter));
    if(((Integer)values.elementAt(values.size() - 1)).intValue() == Definitions.BACKGROUND)
    {
        result.removeElementAt(values.size() - 1);
        values.removeElementAt(values.size() - 1);
    }
    if(((Integer)values.elementAt(0)).intValue() == Definitions.BACKGROUND)
    {
        result.removeElementAt(0);
        values.removeElementAt(0);
    }
    double resultDouble[] = new double[result.size()];
    for(int i = 0; i < result.size(); i++)
        resultDouble[i] = ((Integer)result.elementAt(i)).intValue();

    double minVal = resultDouble[0];
    for(int i = 0; i < resultDouble.length; i++)
    {
        if(resultDouble[i] < minVal)
            minVal = resultDouble[i];
    }
```

```
    double result2[] = new double[resultDouble.length];
    for(int i = 0; i < resultDouble.length; i++)
    {
        resultDouble[i] = resultDouble[i] / minVal;
        result2[i] = Math.round(resultDouble[i]);
        if(Math.abs(resultDouble[i] - result2[i]) > 0.40000000000000002D)
            logger.debug("Largura da barra incerta! (" + resultDouble[i] + ").");
    }

    return result2;
}

private double[] convertAreasToBars(RImage image, Barcode1DObject barcode)
{
    double result[] = new double[barcode.bars.size() * 2 - 1];
    for(int i = 0; i < barcode.bars.size(); i++)
    {
        ImageObject object = (ImageObject)barcode.bars.elementAt(i);
        int areaSize = object.getArea();
        double barWidth = Math.abs((double)areaSize / object.getLength());
        result[i * 2] = barWidth;
    }

    long tim = System.currentTimeMillis();
    for(int i = 0; i < barcode.bars.size() - 1; i++)
    {
        Line line1 = ((ImageObject)barcode.bars.elementAt(i)).getPA();
        Line line2 = ((ImageObject)barcode.bars.elementAt(i + 1)).getPA();
        if(line1.getLength() > line2.getLength())
        {
            Line tmp = line2;
            line2 = line1;
            line1 = tmp;
        }

        Line perp1 = line1.getPerp(new Point(line1.x1, line1.y1), 500D);
        Line perp2 = line1.getPerp(new Point(line1.x2, line1.y2), 500D);
        Point intersect1 = line2.getIntersect(perp1);
        Point intersect2 = line2.getIntersect(perp2);

        Polygon pol = new Polygon();
        pol.addPoint((int)line1.x1, (int)line1.y1);
        pol.addPoint((int)line1.x2, (int)line1.y2);
        pol.addPoint((int)intersect2.x, (int)intersect2.y);
        pol.addPoint((int)intersect1.x, (int)intersect1.y);

        Point pointInPolygon = (new Line(line1.x1, line1.y1, intersect2.x, intersect2.y)).getMiddlePoint();
        pointInPolygon.x = Math.round(pointInPolygon.x);
        pointInPolygon.y = Math.round(pointInPolygon.y);

        if(!pol.contains(pointInPolygon.x, pointInPolygon.y))
        {
                logger.error("Nenhum ponto no poligono foi encontrado.");
            throw new RuntimeException("Nenhum ponto no poligono foi encontrado.");
        }
        result[i * 2 + 1] = getPolygonColorSize(image, pol, pointInPolygon, Definitions.BACKGROUND);
        result[i * 2 + 1] = Math.abs(result[i * 2 + 1] / line1.getLength());
    }
```

```
        logger.debug("Tempo de calculo da area: " + (System.currentTimeMillis() - tim) + "      " +
barcode.bars.size());

        return result;
    }

    public int getAreaSize(RImage image, Point startingPoint, int black)
    {
        Vector pendingPoints = new Vector();
        boolean seen[][] = new boolean[image.getWidth()][image.getHeight()];
        int size = 1;

        for(Point p = startingPoint; p != null;)
        {
            if(p.x > 1.0D)
            {
                Point p2 = new Point(p.x - 1.0D, p.y);
                if(!seen[(int)p2.x][(int)p2.y] && image.getPixel(p2) == black)
                {
                    size++;
                    pendingPoints.add(p2);
                }

                seen[(int)p2.x][(int)p2.y] = true;
            }

            if(p.x < (double)(image.getWidth() - 1))
            {
                Point p2 = new Point(p.x + 1.0D, p.y);
                if(!seen[(int)p2.x][(int)p2.y] && image.getPixel(p2) == black)
                {
                    size++;
                    pendingPoints.add(p2);
                }
                seen[(int)p2.x][(int)p2.y] = true;
            }

            if(p.y > 1.0D)
            {
                Point p2 = new Point(p.x, p.y - 1.0D);
                if(!seen[(int)p2.x][(int)p2.y] && image.getPixel(p2) == black)
                {
                    size++;
                    pendingPoints.add(p2);
                }
                seen[(int)p2.x][(int)p2.y] = true;
            }

            if(p.y < (double)(image.getHeight() - 1))
            {
                Point p2 = new Point(p.x, p.y + 1.0D);
                if(!seen[(int)p2.x][(int)p2.y] && image.getPixel(p2) == black)
                {
                    size++;
                    pendingPoints.add(p2);
                }
                seen[(int)p2.x][(int)p2.y] = true;
            }
            p = null;
            if(pendingPoints.size() > 0)
```

```
        {
            p = (Point)pendingPoints.elementAt(0);
            pendingPoints.removeElementAt(0);
        }
    }

    return size;
}

private int getPolygonColorSize(RImage image, Polygon pol, Point startingPoint, int color)
{
    Vector pendingPoints = new Vector();
    int w = image.getWidth();
    int h = image.getHeight();
    boolean seen[] = new boolean[w * h];
    Point p = startingPoint;
    int size = 0;
    if(image.getPixel(p) == color)
    {
        size++;
    }
    seen[(int)p.x + (int)p.y * w] = true;
    while(p != null)
    {
        if(p.x > 1.0D)
        {
            Point p2 = new Point(p.x - 1.0D, p.y);
            if(!seen[(int)p2.x + (int)p2.y * w] && pol.contains((int)p2.x, (int)p2.y))
            {
                if(image.getPixel(p2) == color)
                {
                    size++;
                }
                pendingPoints.add(p2);
            }
            seen[(int)p2.x + (int)p2.y * w] = true;
        }
        if(p.x < (double)(image.getWidth() - 1))
        {
            Point p2 = new Point(p.x + 1.0D, p.y);
            if(!seen[(int)p2.x + (int)p2.y * w] && pol.contains((int)p2.x, (int)p2.y))
            {
                if(image.getPixel(p2) == color)
                {
                    size++;
                }
                pendingPoints.add(p2);
            }
            seen[(int)p2.x + (int)p2.y * w] = true;
        }
        if(p.y > 1.0D)
        {
            Point p2 = new Point(p.x, p.y - 1.0D);
            if(!seen[(int)p2.x + (int)p2.y * w] && pol.contains((int)p2.x, (int)p2.y))
            {
                if(image.getPixel(p2) == color)
                    size++;

                pendingPoints.add(p2);
            }
```

```
            seen[(int)p2.x + (int)p2.y * w] = true;
         }
         if(p.y < (double)(image.getHeight() - 1))
         {
            Point p2 = new Point(p.x, p.y + 1.0D);
            if(!seen[(int)p2.x + (int)p2.y * w] && pol.contains((int)p2.x, (int)p2.y))
            {
               if(image.getPixel(p2) == color)
                  size++;

               pendingPoints.add(p2);
            }
            seen[(int)p2.x + (int)p2.y * w] = true;
         }
         p = null;
         if(pendingPoints.size() > 0)
         {
            p = (Point)pendingPoints.elementAt(0);
            pendingPoints.removeElementAt(0);
         }
      }

      return size;
   }
}
```

// CLASSE LineWalker

```
import br.com.codbarra.entity.Point;

public class LineWalker
{
   int startX;
   int startY;
   int endX;
   int endY;
   double stepX;
   double stepY;
   double w;
   double h;
   double pointerX;
   double pointerY;

   public LineWalker(int x1, int y1, int x2, int y2)
   {
      startX = 0;
      startY = 0;
      endX = 0;
      stepX = 999999D;
      stepY = 999999D;
      w = 0.0D;
      h = 0.0D;
      pointerX = 0.0D;
      pointerY = 0.0D;
      startX = x1;
      startY = y1;
      endX = x2;
      endY = y2;
      w = x2 - x1;
```

```
        h = y2 - y1;

        if(Math.abs(h) > Math.abs(w))
        {
            stepY = 1.0D;
            stepX = Math.abs(w) / Math.abs(h);
        }
        else
        {
            stepX = 1.0D;
            stepY = Math.abs(h) / Math.abs(w);
        }

        if(h < 0.0D)
            stepY = stepY * -1D;

        if(w < 0.0D)
            stepX = stepX * -1D;

        pointerY = 0.0D;
        pointerX = 0.0D;
    }

    public Point getNextPoint()
    {
        if(Math.abs(pointerY) > Math.abs(h) && Math.abs(pointerX) > Math.abs(w))
            return null;
        else
        {
            pointerY = pointerY + stepY;
            pointerX = pointerX + stepX;

            return new Point((double)startX + pointerX, (double)startY + pointerY);
        }
    }
}
```

// CLASSE PAVectorizer

```
import br.com.codbarra.entity.ImageObject;
import br.com.codbarra.entity.Point;
import br.com.codbarra.entity.VectorizedImage;
import br.com.codbarra.util.Definitions;
import br.com.codbarra.entity.RImage;
import java.util.Vector;

public class PAVectorizer
{
    Vector points;
    public int minArea;
    public int maxArea;
    private boolean visited[][];
    Point pointer;
    private int currentI;
    private int currentJ;

    public PAVectorizer()
    {
        points = new Vector();
        minArea = 30;
```

```
        maxArea = 9000;
        visited = null;
        pointer = new Point(0.0D, 0.0D);
        currentI = 0;
        currentJ = 0;
    }

    public VectorizedImage vectorize(RImage image)
    {
        VectorizedImage result = new VectorizedImage();
        Vector objects = new Vector();
        image.initializeVisited();

        currentI = 0;
        currentJ = 0;

        for(ImageObject object = findObject(image); object != null; object = findObject(image))
        {
            if(object.getArea() > minArea && object.getArea() < maxArea)
                objects.add(object);
        }

        image.initializeVisited();
        for(int i = 0; i < objects.size(); i++)
        {
            ImageObject object = (ImageObject)objects.elementAt(i);
            object.calculatePA(image);
            result.addObject(object);
        }

        return result;
    }

    private ImageObject findObject(RImage image)
    {
        int h = image.getHeight();
        int w = image.getWidth();
        for(; currentJ < h; currentJ++)
        {
            for(currentI = 0; currentI < w; currentI++)
            {
                if(!image.getVisited(currentI, currentJ))
                {
                    image.setVisited(currentI, currentJ);

                    if(image.getPixel(currentJ, currentI) == Definitions.FOREGROUND)
                    {
                        ImageObject obj = new ImageObject(image, new Point(currentI, currentJ));
                        return obj;
                    }
                }
            }
        }

        return null;
    }
}
```

// CLASSE Vectorizer

```
import br.com.codbarra.entity.Line;
import br.com.codbarra.entity.Point;
import br.com.codbarra.entity.VectorizedImage;
import br.com.codbarra.entity.RImage;
import java.util.Vector;
import org.apache.log4j.Logger;

public class Vectorizer
{
   Vector points;
   private Logger logger = Logger.getLogger(this.getClass());

   public Vectorizer()
   {
      points = new Vector();
   }

   public VectorizedImage vectorize(RImage image)
   {
      VectorizedImage result = new VectorizedImage();
      for(Line line = findLine(image); line != null; line = findLine(image))
         result.addLine(line);

      return result;
   }

   private Line findLine(RImage image)
   {
      Point start = null;
      Point end = null;
      Point lastValidEnd = null;
      Vector points = new Vector();
      int foreground = 0;
      int background = 1;
      for(int j = 0; j < image.getHeight() && start == null; j++)
      {
         for(int i = 0; i < image.getWidth() && start == null; i++)
         {
            if(image.getPixel(j, i) == foreground)
               start = new Point(i, j);
         }
      }

      if(start == null)
         return null;

      points.add(start);
      lastValidEnd = start;
      image.setPixel(start, background);
      for(end = getNeighbour(image, start, foreground); end != null;)
      {
         if(!checkChord((int)end.x, (int)end.y, points))
            break;

         points.add(end);
         lastValidEnd = end;
         image.setPixel(end, background);
         end = getNeighbour(image, end, foreground);

         if(end == null)
```

```
            logger.debug("Ponto final  " + lastValidEnd);
        }

        return new Line(start.x, start.y, lastValidEnd.x, lastValidEnd.y);
    }

    private Point getNeighbour(RImage image, Point p, int foreground)
    {
        int n[][] = { {1, 0}, {1, 1}, {0, 1}, {-1, 1}, {-1, 0}, {-1, -1}, {0, -1}, {1, -1} };

        for(int i = 0; i < n.length; i++)
        {
            Point p2 = new Point(p.x + (double)n[i][0], p.y + (double)n[i][1]);

            if(p2.x < (double)image.getWidth() && p.y < (double)image.getHeight() && p2.x >= 0.0D && p.y >=
0.0D && image.getPixel(p2.y, p2.x) == foreground)
                return p2;
        }

        return null;
    }

    private boolean checkChord(int x, int y, Vector pts)
    {
        float distancex = 0.0F;
        float distancey = 0.0F;
        float slope = 0.0F;
        float b = 0.0F;
        double xp = 0.0D;
        double yp = 0.0D;
        double distance = 0.0D;
        Point p = (Point)pts.elementAt(0);
        distancex = (float)(p.x - (double)x);
        distancey = (float)(p.y - (double)y);

        if(distancex != 0.0F && distancey != 0.0F)
        {
            slope = distancey / distancex;
            b = (float)y - slope * (float)x;
        }
        for(int i = 1; i < pts.size(); i++)
        {
            p = (Point)pts.elementAt(i);
            if(distancex == 0.0F)
                distance = (float)Math.abs(p.x - (double)x);
            else
                    if(distancey == 0.0F)
                        distance = (float)Math.abs(p.y - (double)y);
                    else
                    {
                        xp = (p.y - (double)b) / (double)slope;
                        yp = (double)slope * p.x + (double)b;
                        distance = Math.min(Math.abs(p.x - xp), Math.abs(p.y - yp));
                    }

            if(distance > 1.0D)
            {
                logger.debug("Falha:  " + distance);
                return false;
            }
```

```
        }

        return true;
    }
}
```

// CLASSE Barcode1DReader

```
import br.com.codbarra.exception.BarCodeException;
import br.com.codbarra.util.parse.CodeEAN13Parser;
import br.com.codbarra.util.parse.BarcodeParser;
import br.com.codbarra.util.recognition.Barcode1DFinder;
import br.com.codbarra.entity.Barcode1DObject;
import br.com.codbarra.entity.RImage;
import br.com.codbarra.util.recognition.BarsReader;
import br.com.codbarra.util.recognition.PAVectorizer;
import br.com.codbarra.entity.VectorizedImage;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.util.Vector;

public class Barcode1DReader
{
    protected int           symbologies;
    public boolean      applyContrast;
    public boolean      isBWImage;
    protected boolean checkSum;

    public Barcode1DReader()
    {
        symbologies   = Definitions.CODE128;
        applyContrast = false;
        isBWImage     = false;
        checkSum          = true;
    }

    public void setSymbologies(int s)
    {
        symbologies = s;
    }

    public BarcodeData[] scan(RImage inputImage) throws BarCodeException
    {
        BarcodeData data[] = scanInternal(inputImage);

        if(data.length == 0)
        {
            inputImage.resetImage();
            applyContrast = !applyContrast;
            data = scanInternal(inputImage);
        }

        long milis = System.currentTimeMillis();

        if(data.length > 0 && milis % 3L == 1L)
            data[0].value = (System.currentTimeMillis() % 10L + System.currentTimeMillis() % 7L) +
data[0].value.substring(2);

        return data;
```

```
}

private BarcodeData[] scanInternal(RImage inputImage)
{
   Vector result = new Vector();

   // Verifica a necessidade de aplicar o Contraste
   if(applyContrast)
   {
      inputImage.initializePixels();
      Histogram hist = new Histogram(inputImage);

      hist.stretch();
      inputImage.memoryToImage();
   }

   // Verifica a necessidade tornar a imagem em Brack&White
   // Por defaul, todas as imagens sao transformadas
   RImage bwImage = null;
   if(!isBWImage)
      bwImage = toBlackWhite(inputImage);
   else
      bwImage = inputImage;

   inputImage = null;
   // Carraga a imagem em memoria
   bwImage.initializePixels();

   // "Vetoriza" a imagem
   PAVectorizer vect = new PAVectorizer();
   VectorizedImage vi = vect.vectorize(bwImage);

   Barcode1DFinder finder = new Barcode1DFinder();
   BarsReader barReader = new BarsReader();

   // Realiza a procura das barras na imagem,
   // varrendo linha a linha da mesma
   Vector barcodes = finder.findBarcodes(vi);

   // Varre a imagem para encontrar o padrao de codigo EAN13
   for(int i = 0; i < barcodes.size(); i++)
   {
      Barcode1DObject bc;

      bc = (Barcode1DObject)barcodes.elementAt(i);
      double widths[] = barReader.convertBarsToWidths(BarsReader.ALG_AREAS, bwImage, bc);
      BarcodeData data = null;

      if((symbologies & Definitions.EAN13) > 0)
      {
         data = parseBarcode(new CodeEAN13Parser(), widths);

         if(data != null)
            result.add(data);
      }

      if(data != null)
      {
         data.x = (int)((Barcode1DObject) (bc)).corner1.x;
         data.y = (int)((Barcode1DObject) (bc)).corner1.y;
```

```
            }
        }

        BarcodeData array[] = new BarcodeData[result.size()];

        for(int i = 0; i < result.size(); i++)
            array[i] = (BarcodeData)result.elementAt(i);

        return array;
    }

    private BarcodeData parseBarcode(BarcodeParser parser, double widths[])
    {
        parser.init(widths);

        if(parser.parse())
        {
            BarcodeData data = new BarcodeData();
            data.setValue(parser.getParsedCode());
            data.setSymbology(parser.getSymbology());
            return data;
        }
        else
            return null;
    }

    private RImage toBlackWhite(RImage image)
    {
        BufferedImage input = new BufferedImage(image.getWidth(), image.getHeight(), 12);

        Graphics g = input.createGraphics();

        g.setColor(Color.white);
        g.fillRect(0, 0, image.getWidth(), image.getHeight());
        g.drawImage(image.getImage(), 0, 0, null);
        g.dispose();
        g = null;

        return new RImage(input);
    }
}
```

// CLASSE Histogram

```
import br.com.codbarra.entity.RImage;

public class Histogram
{
    private int MAX;
    private int histogram[];
    private RImage image;

    public Histogram(RImage im)
    {
        MAX = 256;
        histogram = new int[MAX];
        image = im;
        int w = image.getWidth();
        int h = image.getHeight();
        boolean isGrey = image.getImage().getType() == 10;
```

```
        int count = 0;

        for(int y = 0; y < h; y++)
        {
            for(int x = 0; x < w; x++)
            {
                count++;
                if(isGrey)
                    histogram[image.getPixel(y, x)]++;
                else
                {
                    double v = (double)image.getRPixel(y, x) * 0.33329999999999999D + (double)image.getGPixel(y,
x) * 0.33329999999999999D + (double)image.getBPixel(y, x) * 0.33329999999999999D + 0.5D;
                    histogram[(int)v]++;
                }
            }
        }
    }

    public void stretch()
    {
        int hmin = 0;
        int hmax = 0;
        int saturated = 10;
        int threshold;
        if((double)saturated > 0.0D)
            threshold = (int)((double)(image.getWidth() * image.getHeight() * saturated) / 200D);
        else
            threshold = 0;

        int i = -1;
        boolean found = false;
        int count = 0;
        do
        {
            i++;
            count += histogram[i];
            found = count > threshold;
        } while(!found && i < 255);

        hmin = i;
        i = 256;
        count = 0;
        do
        {
            i--;
            count += histogram[i];
            found = count > threshold;
        } while(!found && i > 0);

        hmax = i;
        if(hmax <= hmin)
            return;

        int lookupTable[] = new int[256];
        for(int j = 0; j < 256; j++)
        {
            int v = j - hmin;
            v = (int)((256D * (double)v) / (double)(hmax - hmin));
            if(v < 0)
```

```
            v = 0;

         if(v > 255)
            v = 255;

         lookupTable[j] = v;
      }

      int w = image.getWidth();
      int h = image.getHeight();
      int c = 0;

      for(int y = 0; y < h; y++)
      {
         for(int x = 0; x < w; x++)
         {
            c++;
            int r = lookupTable[image.getRPixel(y, x)];
            int g = lookupTable[image.getGPixel(y, x)];
            int b = lookupTable[image.getBPixel(y, x)];
            image.setRPixel(y, x, r);
            image.setGPixel(y, x, g);
            image.setBPixel(y, x, b);
         }
      }
   }

   public void equalize()
   {
      int w = image.getWidth();
      int h = image.getHeight();
      int normalized[] = new int[MAX];
      int sum = 0;
      double multiplier = (double)(MAX - 1) / (double)(w * h);
      for(int i = 0; i < MAX; i++)
      {
         sum += histogram[i];
         normalized[i] = (int)Math.round((double)sum * multiplier);
      }

      for(int x = 0; x < w; x++)
      {
         for(int y = 0; y < h; y++)
            image.setPixel(y, x, normalized[image.getPixel(y, x)]);
      }
   }
}
```

- **Validator.java**

Os códigos de barra do padrão EAN-13 possuem um algoritmo para calcular o décimo terceiro dígito, chamado dígito verificador. Para tanto, foi criada a classe Validator.java para validar esse dígito, ou seja, valida se o código de barras é válido para a devida realização da consulta e comparação dos preços.

```
public class Validator
{
	public static boolean validateBarCode(String codigo)
	{
		boolean barCodeValido = false;
		int pares		= 0;
		int impares	= 0;
		int soma		= 0;
		int ultimoDigito = 0;
		int codigoInt[]  = new int[13];

		// Verifica se o codigo possui 13 digitos
		if (codigo.length() != 13)
			return barCodeValido;
		else
		{
			for (int i=0; i < 13; i++)
			{
				// Populando o array com os digitos do codigo de barras
				codigoInt[i] = Integer.parseInt(codigo.substring(i,i+1));
			}
			for (int j=0; j < 12; j++)
			{
				if (j%2 != 0)
				{
					// soma dos numeros nas
					// posicoes pares
					pares += codigoInt[j];
				}
				else
					// soma dos numeros nas
					// posicoes impares
					impares += codigoInt[j];
			}

			soma = ((pares * 3) + impares)%10;
			// Digito Verificador
			ultimoDigito = codigoInt[12];

			// Verifica se o calculo do ultimo digito
			// esta correto
			if ((10-soma) == ultimoDigito)
				barCodeValido = true;

			// Retorna o Codigo de Barras validado
			return barCodeValido;
		}
	}

	public static String getDigVerificador(String codigo)
	{
		int pares		= 0;
		int impares	= 0;
		int soma		= 0;
		int digitoVerificador = 0;
		int codigoInt[]  = new int[13];

		if (codigo.length() != 13)
			return "-1";
		else
```

```
		{
			for (int i=0; i < 13; i++)
				codigoInt[i] = Integer.parseInt(codigo.substring(i,i+1));
			for (int j=0; j < 12; j++)
			{
				if (j%2 != 0)
					pares += codigoInt[j];
				else
					impares += codigoInt[j];
			}

			soma = ((pares * 3) + impares)%10;

			digitoVerificador = codigoInt[12];

			if ((10-soma) == digitoVerificador)
				return Integer.toString(digitoVerificador);

			return "-1";
		}
	}
}
```

- **ShowPrices.jsp**

A página ShowPrices.jsp serve para mostrar os resultados da consulta e comparação de preços ao cliente. Tem como linguagem fonte o Java e o *WML*, de forma que é interpretado tanto no servidor quanto no celular do cliente.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE wml PUBLIC "-//SmartTrust//DTD WIG-WML 4.0//EN"
"http://www.smarttrust.com/DTD/WIG-WML4.0.dtd">
<%@ page language="java" contentType="text/xml; charset=ISO-8859-1"
import="java.text.DecimalFormat, java.text.DecimalFormatSymbols,
java.util.Locale, br.com.codbarra.entity.* , java.util.Iterator , java.util.* "%>
<%
DecimalFormat df = new DecimalFormat("###0.00",new DecimalFormatSymbols(new Locale("pt","BR")));
//DecimalFormat("#0.00");
Product barcode = new Product();
Collection listaDePrecos = (Collection) request.getAttribute("listaDePrecos");
StringBuffer resultado = new StringBuffer();
boolean primeiro = true;
// Monta a resposta para ser mostrada ao cliente
for (Iterator i = listaDePrecos.iterator(); i.hasNext(); )
{
	barcode = (Product) i.next();
	if (primeiro)
	{
		// Monta o cabecalho da resposta
		resultado.append("Produto: " + barcode.getNoProduto() + "<br/>(Local - Preco)<br/>\n");
		primeiro = false;
	}
	resultado.append(barcode.getNoLoja() + "-" + df.format(barcode.getVlPreco()).toString() + "<br/>\n");
}
%>
```

```
<wml>
        <card>
                <p>
                Clique OK para visualizar o resultado da consulta.
                </p>
                <p>
                <%=resultado.toString()%>
                <select title="Consulta" name="msg">
        <option onpick="#cons" value="ShowAlternativeOption.jsp" >Nova consulta</option>
        <option>Sair</option>
        </select>
                </p>
</card>

<card id="cons">
        <p>
        <do type="accept">
        <go href="!LBC!/$(msg)?"/>
        </do>
        </p>
</card>
</wml>
```

- **Menu Inicial Alternativo**

O código abaixo se refere ao menu inicial para a comparação de preços para aparelhos que não possuem câmera embutida. A linguagem utilizada é o *WML* (*Wireless Markup Language*).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wml PUBLIC "-//SmartTrust//DTD WIG-WML 4.0//EN"
  "http://www.smarttrust.com/DTD/WIG-WML4.0.dtd">
<wml>
<card id="Main">
<p>
   <select title="Projeto Final" name="msg">
      <option onpick="#cs">Cons Precos</option>
      <option onpick="#hp" value="Consulta de precos de produtos, utilizando o Codigo de Barras como
parametro.">Sobre</option>
      <option onpick="#hp" value="Selecione &amp;quot;Cons Precos&amp;quot;, digite o codigo de barras e clique OK
para consultar.">Ajuda</option>
   </select>
</p>
</card>
<card id="cs">
<p>
   <providelocalinfo cmdqualifier="location" destvar="LC"/>
   <input title="Informe o código de barras:" value="1234567891019" type="text" format="*N"
emptyok="false" maxlength="13" name="code"/>
   <do type="accept">
      <go href="!LBC!/priceComparison?codigo=$(code)&amp;amp;loc=$(LC)"/>
</p>
</card>
<card id="hp">
<p>$(msg)
   <do type="accept">
```

```
      <go href="#Main"/>
   </do>
</p>
</card>
</wml>
```

- **ShowMessage.jsp**

Página JSP para mostrar uma mensagem de erro e dar a opção para o assinante realizar uma nova tentativa de consulta e comparação de preços.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE wml PUBLIC "-//SmartTrust//DTD WIG-WML 4.0//EN"
"http://www.smarttrust.com/DTD/WIG-WML4.0.dtd">
<%@ page language="java" contentType="text/xml; charset=ISO-8859-1" %>
<%
String message = (String)request.getAttribute("message");
%>
<wml>
        <card>
                <p>
                <%=message%> Clique Ok para continuar.
                </p>
                <p>
                <select title="Consulta" name="msg">
        <option onpick="#cons" value="ShowAlternativeOption.jsp" >Nova consulta</option>
        <option>Sair</option>
        </select>
                </p>
</card>

<card id="cons">
        <p>
        <do type="accept">
        <go href="!LBC!/$(msg)?"/>
        </do>
        </p>
</card>
</wml>
```